PERBANDINGAN SISTEM KENDALI *AR-DRONE* MENGGUNAKAN *LEAP MOTION* PADA *NODE JS* DAN *LABVIEW*

SKRIPSI

KEMINATAN TEKNIK KOMPUTER

Untuk memenuhi sebagian persyaratan
memperoleh gelar Sarjana Komputer

Disusun oleh:
Hendra
NIM: 135150301111133

PROGRAM STUDI TEKNIK INFORMATIKA
JURUSAN TEKNIK INFORMATIKA
FAKULTAS ILMU KOMPUTER
UNIVERSITAS BRAWIJAYA
MALANG
2018

PENGESAHAN

Perbandingan Sistem kendali *Ar-Drone* Menggunakan *Leap motion* Pada *Node js* Dan *Labview*

SKRIPSI

Diajukan untuk memenuhi sebagian persyaratan
memperoleh gelar Sarjana Komputer

Disusun Oleh :
Hendra
NIM: 135150301111133

Skripsi ini telah diuji dan dinyatakan lulus pada
6 Agustus 2018
Telah diperiksa dan disetujui oleh:

Dosen Pembimbing I	Dosen Pembimbing II
Gembong Edhi Setyawan, S.T, M.T	Wijaya Kurniawan, S.T, M.T
NIK: 201208 761201 1 001	NIP: 357301 250182 0 003

Mengetahui
Ketua Jurusan Teknik Informatika

Tri Astoto Kurniawan, S.T., M.T., Ph.D.
NIP: 19710518 200312 1 001

PERNYATAAN ORISINALITAS

Saya menyatakan dengan sebenar-benarnya bahwa sepanjang pengetahuan saya, di dalam naskah skripsi ini tidak terdapat karya ilmiah yang pernah diajukan oleh orang lain untuk memperoleh gelar akademik di suatu perguruan tinggi, dan tidak terdapat karya atau pendapat yang pernah ditulis atau diterbitkan oleh orang lain, kecuali yang secara tertulis disitasi dalam naskah ini dan disebutkan dalam daftar pustaka.

Apabila ternyata didalam naskah skripsi ini dapat dibuktikan terdapat unsur-unsur plagiasi, saya bersedia skripsi ini digugurkan dan gelar akademik yang telah saya peroleh (sarjana) dibatalkan, serta diproses sesuai dengan peraturan perundang-undangan yang berlaku (UU No. 20 Tahun 2003, Pasal 25 ayat 2 dan Pasal 70).

Malang, 6 Agustus 2018

Hendra

NIM: 135150301111133

KATA PENGANTAR

Puji syukur kehadirat Allah SWT yang telah melimpahkan rahmat, taufik dan hidayah-Nya sehingga laporan skripsi yang berjudul "Perbandingan Sistem Kendali *Ar-Drone* Menggunakan *Leap Motion* Pada *Node Js* Dab *Labview*" ini dapat terselesaikan. Laporan skripsi ini disusun dalam rangka memenuhi persyaratan untuk memperoleh gelar Sarjana Komputer di Fakultas Ilmu Komputer

Penulis menyadari bahwa penyusunan laporan skripsi ini tidak lepas dari bantuan berbagai pihak baik secara langsung maupun tidak langsung. Dalam kesempatan ini penulis ingin menyampaikan ucapan terima kasih atas bantuannya kepada semua pihak, sehingga penulis dapat menyelesaikan laporan ini dengan baik. Ucapan terima kasih tersebut khususnya kepada :

1. Allah yang Maha Esa yang selalu memberikan petunjuk dan hikmah dalam penulisan ini.
2. Orang Tua dan Keluarga atas nasehat, kasih sayang serta dukungan materil dan moril.
3. Gembong Edhi Setyawan, S.T, M.T dan Wijaya Kurniawan, S.T, M.T selaku dosen pembimbing yang telah memberikan dukungan, saran dan ilmu dalam membimbing penulis hingga pengerjaan skripsi ini selesai.
4. Bapak Wayan Firdaus Mahmudy, S.Si, M.T, Ph.D. selaku Dekan Fakultas Ilmu Komputer Universitas Brawijaya Malang.
5. Bapak Heru Nurwarsito, Ir., M.Kom. selaku Wakil Dekan I Bidang Akademik Fakultas Ilmu Komputer Universitas Brawijaya Malang.
6. Bapak Sabriansyah Rizqika Akbar, S.T, M.Eng. selaku Ketua Program Studi Teknik Komputer Universitas Brawijaya Malang.
7. Tri Astoto Kurniawan, S.T., M.T., Ph.D. selaku ketua jurusan Teknik Informatika.
8. Seluruh civitas akademika Informatika Universitas Brawijaya dan teman-teman Teknik Komputer Angkatan 2013 yang telah banyak memberi bantuan dan dukungan selama peneliti menempuh studi di Teknik Komputer Universitas Brawijaya dan selama penyelesaian skripsi ini.
9. Ayang Setiyo Putri, Achmad Baecuni, Dimas Angger, Sabita Wildani , Yusril Dewantara, Andyan Bina, Sabitha Wildani, Amroy Casro, Fajar Miftakhul selaku teman seangkatan yang telah memberikan ilmu terkait penelitian yang pernah dilakukan dan teman seperjuangan *quadcopter* Haqqi, Enno, Muliya, serta rekan-rekan lainnya yang turut memberikan motivasi dan do'a hingga pengerjaan skripsi ini selesai.

10. Dan orang-orang yang selalu mendukung serta mendoakan kelancaran proses skripsi ini yang tidak bisa disebutkan satu per satu atas semua doa dan dukungannya.

Dengan segala keterbatasan pengetahuan yang dimiliki, penulis sadar bahwa penulisan laporan skripsi ini masih jauh dari kesempurnaan. Sehingga penulis berharap agar laporan skripsi ini dapat bermanfaat dan merupakan salah satu informasi yang berguna bagi pembaca.

Malang, 6 Agustus 2018

Penulis

hend11draa@gmail.com

ABSTRAK

Saat ini teknologi sangatlah berkembang dari berbagai aspek salah satunya teknologi pesawat tanpa awak. Dalam teknologi pesawat tanpa awak dikenalah *quadcopter*. *Quadcopter* merupakan sebuah mesin terbang yang berfungsi dengan kendali jarak jauh oleh pilot mampu mengendalikan dirinya sendiri menggunakan hukum aerodinamika untuk mengangkat dirinya atau muatan lainnya. Untuk saat ini pengontrolan *quadcopter* mayoritas masih menggunakan sebuah *remote* pengontrol *quadcopter* berupa *joystick* atau menggunakan aplikasi berbasis *android*. Padahal sebuah gerakan sederhana dalam pengontrolan *quadcopter* dapat mempermudah pilot untuk mengendalikan *quadcopter* tersebut. Munculnya inovasi dari *Natural User Interfaces (NUI)* yaitu untuk menggunakan bahasa alamiah manusia seperti suara, gerakan, ataupun pandangan untuk berkomunikasi dengan teknologi *quadcopter*. Dalam sistem pada penelitian ini dibuat kendali yang memanfaatkan inovasi dari *NUI* untuk mengendalikan *quadcopter* berupa gerakan tangan sederhana dari pengguna. Gerakan tangan pengguna akan dideteksi menggunakan *leap motion*. *Leap motion* adalah sebuah *device* yang mana terdapat sensor optik dan cahaya in*frame*rah untuk mendeteksi gerak-gerik suatu tangan. *Leap motion* dan *quadcopter* diprogram menggunkan *javascript* dan *labview.* Setelah melakukan pengujian ketepatan gerakan dan kecepatan, dihasilkan persentase ketepatan gerakan sebesar 100%. Pada kecepatan yang dihasilkan oleh *quadcopter* untuk gerakan *pitch*, *roll*, *yaw*, *gaz* yaitu berbanding lurus dengan nilai yang diperoleh dari gerakan pengguna, hal ini menandakan saat *input* dari pengguna semakin besar, maka *quadcopter* akan bertambah cepat dan begitu pula sebaliknya. Dari pengujian *delay* sistem yang didapat dari pengguna menggerakkan tangannya hingga *quadcopter* bergerak sesuai dengan intruksi pengguna menghasilkan *delay* sebesar 0,258 detik sedangkan *delay response* sistem yang dilakukan pada *labview* menghasilkan 0,131 detik dari hasil tersebut menunjukkan bahwa pemrograman *labview* lebih responsif digunakan untuk pengontrolan *quadcopter* dibandingkan menggunakan bahasa pemrograman *javascript*.

Kata kunci: *Quadcopter*, *Leap motion, Javascript, Labview,Gerakan tangan, Natural User Interfaces*

ABSTRACT

Technology nowadays is developed in various aspects, one of them is unmanned aircraft technology. In unmanned aircraft technology introduced the quadcopter. Quadcopter is a flying machine that works with the remote control that controled by the pilot using the law of aerodynamics to lift himself or other object. The majority of quadcopter controls still using a remote controller in the form of a joystick or using android based applications. Whereas a simple movement in quadcopter control can help the pilot to control the quadcopter properly. The innovations from Natural User Interfaces (NUI) is to use natural human language such as voice, movement, or view to communicate with quadcopter. In this research will created a control that using innovation from NUI to control quadcopter in the form of simple hand movement of the user. User's hand movement will be detected using leap motion. Leap motion is a device where optical sensor and infrared light detecting the movements of a hand. Leap motion and quadcopter are programmed using javascript and labview. After testing the performance of its precision the movement and speed, the result percentage of motion accuracy is 100%. At the resulst speed that generated by quadcopter to pitch, roll, yaw, gaz moves are directly proportional with value obtained from user movement, When the input from the user gets larger, the quadcopter speed will getting faster and otherwise. The result of delay response test that obtained from the user moving his hands using node.js is 0.258 seconds while the result of delay response system that performed using labview is 0.131 seconds. Based on its resulst that using labview programming to control the quadcopter is more responsive than using programming languages javascript.

Keyword: *Quadcopter*, *Leap motion*, *Javascript, Labview, Hand tracking, Natural User Interfaces*

DAFTAR ISI

DAFTAR TABEL

DAFTAR GAMBAR

PENDAHULUAN

1.1 Latar belakang

Teknologi pesawat terbang saat ini sangatlah berkembang. Salah satu contoh dari teknologi itu ialah pesawat tanpa awak atau yang lebih dikenal dengan *drone*. *Drone* merupakan pesawat tanpa awak yang cara pengoperasiannya bisa menggunakkan pilot dengan mengontrolnya dari jarak jauh maupun tanpa pilot yang menggunakan hukum *aerodinamis* untuk mengangkat badan dan muatan yang ada pada *drone* itu sendiri. Contoh dari salah satu macam *drone* ialah *quadcopter* yaitu *drone* yang memiliki 4 buah baling-baling pada badan *drone*. Penerapan *quadcopter* sangatlah bervariasi mulai digunakan sebagai alat pemantau, bencana alam, kemacetan lalu lintas pengambilan gambar pemandangan maupun sebagai alat militer. Sebagai contoh dari Damar (2017) *drone* atau *quadcopter* digunakan dalam misi pemadaman kebakaran membantu memetakan lokasi pencarian korban. Dikarenakan dalam *drone* telah dilengkapi kamera, teknologi pencitraan suhu, serta sensor gas yang digunakan untuk membantu misi pemadam kebakaran.

Menurut Sarkar, et al., (2016) tentang pengendalian *quadcopter* dengan *leap motion controller*. Pengguna akan memberikan perintah berupa gerakan tangan diatas *leap motion*. Data dari *leap motion* akan diolah menggunakan *ROS (Robot Operating System)* kemudain data pergerakan tangan akan dikirim menuju *quadcopter* yang digunakan sebagai acuan perintah gerakan pada *quadcopter*.

Menurut Skraba, et al., (2015) dalam penelitiannya menjelaskan tentang pembuatan *prototype* kursi roda. Pada *prototype* tersebut menggunakan *leap motion*, *keyboard* dan suara sebagai *input* pergerakan kursi roda. *Prototype* menggunakan *arduino uno* sebagai *microcontroller* pengolahan data. *Leap motion* digunakan kontol kursi roda melalui *gesture recognition,* suara melalui *speech recognition* sedangkan *keyboard* digunakan untuk *input*an manual pada *prototype*. Pada *prototype* menggunakan bahasa pemrograman *node.js* yang digunakan sebagai penghubung *leap motion* dan *arduino uno*. Hal tersebut menunjukkan bahwa *leap motion* dapat digunakan menggunakan bahasa pemrograman *node.js*.

Menurut Tolentino, et al., (2016) dalam penelitian ini menjelaskan tentang penggunaan *leap motion* sebagai pengganti *gloves sensor* yang dianggap memiliki banyak kerugian dalam pemakaiannya. *Gloves sensor* sendiri merupakan sensor berbentuk sarung tangan yang digunakan untuk menggerakan tangan robot. Penerapan *gloves sensor* dianggap memiliki banyak kekurangan yaitu membutuhkan kalibrasi pada saat penggunaannya dan disetiap tangan yang memakai berbeda *gloves sensor* membutuhkan kalibrasi lagi. Sehingga dalam penelitiannya peneliti menggukan *leap motion* sebagai pengganti yang diterapkan menggukan *labview* sebagai bahasa pemrograman. Penelitian ini menunjukkan bahwa *leap motion* dapat berkomunikasi dengan *labview*.

Menurut Ardhana, et al., (2018) tentang pengendalian *quadcopter* dengan menggunakan *leap motion* diprogram *node js* menunjukkan hasil yang berbanding lurus pada saat percobaan gerakan *pitch*, *roll*, *gaz* dimana saat *input* dari pengguna semakin besar, maka *quadcopter* akan bertambah cepat dan begitu pula sebaliknya. Sedangkan dari pengujian *delay* mendapatkan hasil sebesar 0,258 detik.

Berdasarkan penelitian diatas dibuatlah sebuah penelitian yang mana akan membandingkan pengendalian *quadcopter* menggunakan *leap motion* pada *node js* dan *labview*. Dimana perbandingan berfokus pada bagaimana tingkat ketepatan gerakan, tingkat kecepatan dan tingkat *delay* respon sistem pada *quadcopter*. Pada penelitian akan diterapkan *Natural User Interface (NUI)* sebagai kontrol kendali navigasi sebuah *quadcopter*. Dalam penelitian ini *Natural User Interface (NUI*) yang digunakan yaitu gerakan tangan sederhana yang dilacak oleh *leap motion*.

1.2 Rumusan masalah

Berdasarkan latar belakang maka terdapat rumusan masalah dari penelitian ini adalah sebagai berikut:

1. Bagaimana menentukan nilai koordinat untuk setiap gerakan sistem kendali navigasi *AR Drone* menggunakan *leap motion* pada pemrograman *node js* dan *labview*?
2. Bagaimana tingkat ketepatan gerakan sistem kendali navigasi *AR Drone* menggunakan *leap motion* pada pemrograman *node js* dan *labview*?
3. Bagaimana tingkat *delay respone* sistem kendali navigasi *AR Drone* menggunakan *leap motion* pada pemrograman *node js* dan *labview*?
4. Bagaimana tingkat kecepatan pergerakan *quadcopter* yang dihasilkan oleh *leap motion* sebagai sistem kendali *quadcopter* pada pemrograman *node js* dan *labview*?

1.3 Tujuan

Sesuai dengan rumusan masalah yang telah disebutkan maka terdapat tujuan dari penelitian ini sebagai berikut:

1. Mengetahui nilai koordinat untuk menentukan setiap gerakan sistem kendali navigasi *AR Drone* menggunakan *leap motion* pada pemrograman *node js* dan *labview*.
2. Mengetahui tingkat ketepatan gerakan sistem kendali navigasi *AR Drone*menggunakan *leap motion* pada pemrograman *node js* dan *labview.*
3. Mengetahui seberapa besar tingkat *delay respone sistem* kendali navigasi *AR Drone* menggunakan *leap motion* pada pemrograman *node js* dan *labview.*

4. Mengetahui kecepatan pergerakan *quadcopter* yang dihasilkan oleh *leap motion* sebagai sistem kendali *quadcopter* pada pemrograman *node js* dan *labview.*

1.4 Manfaat

Terdapat manfaat dalam penelitian ini antara lain:

1. Berguna sebagai acuan dalam penelitian sebelumnya yang bertujuan untuk meningkatkan kualiatas semakin baik.
2. Mengetahui hasil dari *delay* respon sistem dari penelitian yang membandingkan *node.js* dan *labview* sebagai bahasa pemrograman untuk mengontrol gerakan *quadcopter*.
3. Mempermudah pengguna pemula yang akan mencoba mengendalikan *quadcopter*.

1.5 Batasan masalah

Penentuan batasan masalah dilakukan agar permasalahan yang sudah dirumuskan dapat lebih terfokus dan tidak meluas. Pada penelitian ini, batasan masalah tersebut antara lain:

1. *Quadcopter* mengunakan *AR Drone Parrot* 2.0.
2. Menggunakan satu telapak tangan sebelah kanan.
3. Penggunaan *AR Drone* dilakukan diruangan *indoor.*
4. Tidak boleh ada tangan lain disekitar tangan pengguna ketika melakukan pengendalikan *quadcopter.*
5. Gerakan *quadcopter* yang dikendalikan yaitu *roll, pitch, gaz, takeoff, hover* dan mendarat.

1.6 Sistematika pembahasan

Sistematika penulisan dalam skripsi ini meliputi sebagai berikut:

BAB 1 Pendahuluan

Bab ini menjelaskan tentang latar belakang, rumusan masalah, tujuan, manfaat, batasan masalah, dan sistematika dalam penulisan.

BAB 2 Landasan Kepustakaan

Bab ini membahas tinjauan pustaka dan dasar teori pada penelitian-penelitian sebelumnya yang mendukung dalam pembuatan sistem ini.

BAB 3 Metodologi

Bab ini menjelaskan tentang langkah-langkah dalam menjalankan penelitian ini, antara lain bagaimana menentukan metode

penelitian, analisis kebutuhan, perancangan dan implementasi sistem, pengujian dan analisis, serta penutup.

BAB 4 Analisis dan Perancangan

Bab ini membahas apa saja kebutuhan pengguna, kebutuhan sistem, fungsional dan non fungsional yang diperlukan dalam merancang sistem.

BAB 5 Implementasi dan Pengujian

Bab ini berisi implementasi dari yang sudah dikonsep pada bagian bab perancangan. Pada tahap ini merupakan inti dari pada penelitian, karena hasil penelitian bisa dilihat saat implementasi dari penelitian yang sudah dilaksanakan, serta pengujian dari perancangan yang sudah mendapatkan hasilnya.

BAB 6 Penutup

Bab ini berisi tentang kesimpulan dari penelitian yang telah dilakukan dan saran dari pengguna untuk pengembangan sistem selanjutnya.

BAB 2 LANDASAN KEPUSTAKAAN

Pada bab ini berisikan studi literatur yang berisikan penjelasan dasar-dasar teori yang bertujuan untuk menunjang penelitian yang akan dilaksanakan. Pada bab ini juga terdapat pembahasan-pembahasan penelitian terdahulu yang berhubungan dengan penelitian yang akan dilakukan penulis.

2.1 Kajian Pustaka

Berikut ini adalah beberapa penelitian tentang *gesture control* menggunakan *leap motion*.

Menurut penelitian Pallas et al., (2017) menjelaskan sistem kendali *quadcopter* menggunakan *input*an suara melalui smartphone. Suara hasil *input*an akan dirubah menjadi kata. Kata tersebut akan dicocokan dengan *database* yang sudah dibuat menggunakan algoritme *stopword removal wordlist*. Hasil dari pencocokan akan dikirim melalui *arduino* dan dilakukan pengolahan data hasilnya berupa *output* gerakan *quadcopter*. Waktu akan terus bertambah dalam melakukan *text preprocessing* dan pengolahan suara seiring dengan banyaknya kata yang diberikan. Pada percobaan pertama masukan 1 kata pengolahan suara menghasilkan *delay* 3 detik. Kemudian pada percobaan kedua dengan memasukan 2 kata pengolahan suara menghasilkan *delay* 4 detik.

Menurut Sarkar et al., (2016) menjelaskan tentang pengont*roll*an *quadcopter* menggunakan *leap motion* dengan gerakan-gerakan tangan sederhana. Data gerakan tangan dikirim melalui ground station berupa kabel usb 2.0. *Ground station* menjalankan *ROS (Robot Operating System)* di *linux* yang digunakan sebagai *opertaing sistem* dalam implementasi. *Python* sebagai bahasa program yang digunakan untuk menghubungkan antara *AR Drone* dengan gerakan tangan.

Menurut Pititeeraphab, et al., (2016) tentang sistem kendali lengan robot menggunakan *leap motion.* Dalam penelitian ini menerapkan prinsip *leap motion* dan kontrol servo motor. Penelitian ini dirancang dan dibangun yang terdiri dari 3 bagian utama yaitu detektor, sinyal kontrol dan pengolahan data menggunakan *microcontroller* yang terdapat pada *arduino UNO R3*, tampilan dari lengan robot. Pada bagian pertama komponen internal dari *leap motion* yaitu sensor kamera digital akan mendeteksi gerakan tangan pengguna untuk mengontrol lengan robot. *CPU* yang terdapat pada *leap motion* akan memproses *visual* lalu mengirimkan data posisi koordinat gerakan tangan dan jari-jari ke komputer. Data tersebut akan ditampilkan secara *3D* dikomputer. Setelah itu data dikirim menuju *microcontroller* untuk pengolahan sinyal dan mengirim perintah kontrol untuk motor servo sebagai kontrol lengan robot.

Menurut Pribadi et al., (2017) penelitian yang dilakukan tentang pengendalian *quadcopter* menggunakan prinsip *virtual reality* dengan *google cardboard. User* memberikan *input*an gerakan kepala, setelah itu *input*an akan di

klasifikasikan menurut jenis gerakan kepala dan dikirimkan menuju aplikasi kendali *quadcopter* untuk dijadikan *input*an. Setelah itu pembuatan tampilan *virtual reality* menggunakan *stereoscopic display*. Kemudian data *input* dicocokkan dengan data pembacaan dari *gyroscope* yang dihasilkan dari gerakan kepala untuk dijadikan *output* gerakan *quadcopter*. Namun pada penelitian ini terdapat beberapa kekurangan pada pembacaan gerakan kepala yang seharusnya bisa membaca 6 gerakan tetapi hanya dapat membaca 4 gerakan saja hal ini dikarenakan *trinus virtual reality (VR)* tidak bisa membaca gerakan kepala miring ke kanan maupun ke kiri.

Menurut Ardhana, et al., (2018) tentang pengendalian *quadcopter* dengan menggunakan *leap motion* diprogram *node js* menunjukkan hasil yang berbanding lurus pada saat percobaan gerakan *pitch*, *roll*, *gaz* dimana saat *input* dari pengguna semakin besar, maka *quadcopter* akan bertambah cepat dan begitu pula sebaliknya. Sedangkan pengujian *delay* dari reponse sistem sebesar 0,258 detik.

Berdasarkan penelitian-penelitian tersebut dapat disimpulkan bahwa banyak cara mengendalikan gerakan *quadcopter*. Pada penelitian (Hadi, Setyawan, & Maulana, 2017), menggendalikan gerakan *quadcopter* menggunkan *microsoft kinect* sebagai alat pendeteksi gerakan manusia namun *microsoft kinect* kekurangan yaitu ukuruan yang terlalu besar masih dirasa kurang praktis untuk penggunaan secara terus menerus. Hal tersebut jauh lebih baik jika kontrollernya menggunakan *leap motion* yang ukurannya lebih kecil dari *microsoft kinect* sehingga menjadi lebih praktis untuk penggunaan kendali *quadcopter*. Selanjutnya pada penelitian (Pribadi, Jonemaro, & Setyawan, 2017) juga terdapat kelemahan yaitu hanya dapat menjalankan 4 dari 6 gerakan kepala yang bisa dimasukan untuk mengendalikan *quadcopter* sehingga ada beberapa gerakan *quadcopter* yang tidak bisa dijalankan hal tersebut tentu akan mempengerahui sistem dimana *quadcopter* yang seharusnya bisa bergerak kenan dan kekiri menjadi tidak bisa dijalankan, sehingga dengan penggunaan *leap motion* diharap seluruh gerakan *quadcopter* bisa diimplementasikan. Sedangkan penelitian yang dilakukan (Pallas, Setyawan, & Prasetio, 2017) dengan menggunakan suara terdapat kelemahan yaitu pada *delay respon sistem* yang sangat lama seiring dengan bertambahnya jumlah kata yang dimasukan oleh pengguna maka semakin bertambah jumlah waktu respon sistemnya, menggunakan *leap motion* hal tersebut dapat diatasi dikarenakan proses deteksi *leap motion* mencatat hanya butuh waktu dibawah 1 detik, tangan yang akan dijadikan *input*, dibandingkan dengan menggunakan suara yang membutuhkan waktu 3 detik dan semakin bertambah ketika jumlah data *input*an semakin banyak. Sedangkan penelitian yang dilakukan oleh (Pititeeraphab, 2016) yang menggunakan *leap motion* sebagai pengendali lengan robot, berbeda dengan (Pititeeraphab, 2016) pada penelitian ini digunakan pada *quadcopter* sebagai objek penelitian. Implentasi NUI menggunakan *leap motion* untuk menggerakkan *quadcopter* sebenarnya sudah pernah dilakukan oleh (Ardhana, Setyawan, & Arwan, 2018) namun penerapan berfokus pada pemrograman *node js*. Sebagai pembeda dari penelitian yang sudah dilakukan (Ardhana, Setyawan, & Arwan,

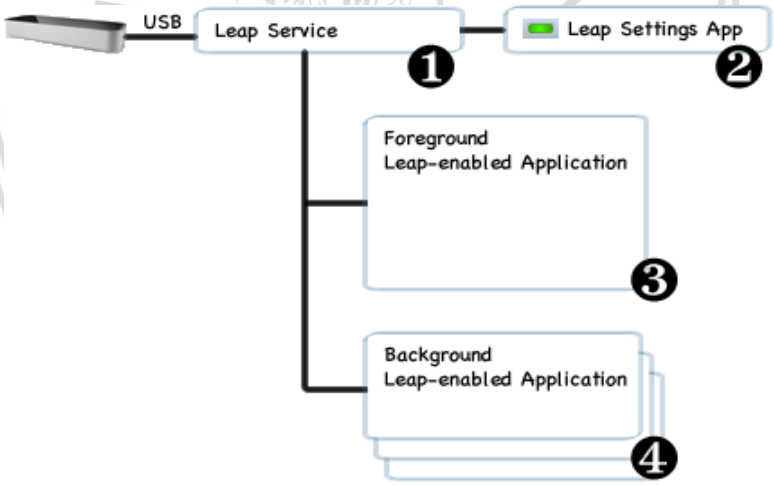
2018) penilitian yang akan penulis lakukan akan berfokus pada perbanding implentasi *NUI* menggunakan *leap motion* untuk menggerakkan *quadcopter* pada pemrograman *node js* yang telah di buat oleh (Ardhana, Setyawan, & Arwan, 2018) dengan implementasi *NUI* menggunakan *leap motion* untuk menggerakkan *quadcopter* pada pemrograman *labview*.

2.2 Dasar Teori

Pada dasar teori ini akan kan dibahas tentang dasar-dasar teori yang berkaitan dengan penelitian yang akan dilakukan penulis seperti *leap motion*, *quadcopter*, *NUI (Natural User Interfaces)*, *labview* serta *node.js*.

2.2.1 Leap Motion

Leap motion merupakan *SDK (Software Development Kit)* dimana *Leap motion SDK* sendiri dapat digunakan untuk pengembangan suatu *software*, sistem komputer, *hardware paltform* maupun *konsol video game*. Pada *Leap motion SDK* sendiri telah disediakan *Library* akan terhubung dengan *Leap motion* dan menyediakan data *tracking* dari aplikasi yang dapat digunakan untuk pengambangan suatu sistem. *Library* ini dapat digunakan pada bahasa pemrograman C++, C#, *Javascript*, *labview* maupun *Python*. Berikut merupakan skema sistem kerja *leap motion* pada sebuah aplikasi terlihat pada Gambar 2.1



Gambar 2.1 Skema *leap motion* dalam sebuah aplikasi

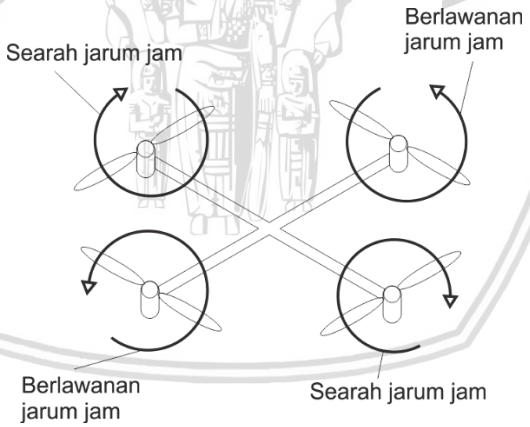
Sumber : (Inc, 2013)

1. *Leap motion* menerima data dari *Leap motion* cont*roll*er yang lewat dari USB.Pada proses ini mengirimkan informasi *Leap-enabled applications* yang menunjukkan aplikasi sedang berjalan. Secara *default*, layanan mengirim data *tracking* tangan untuk *foreground application* berupa gambar tangan seperti tulang dan sendi-sendi pada tangan.
2. Aplikasi *leap motion* dapat kita konfigurasi sendiri untuk penggunaannya tanpa mengikuti layanan ada pada *leap motion* contohnya seperti pada penelitian ini penulis hanya mengambil koordinat untuk dijadikan *input* pada gerakan *quadcopter*.

3. *Leap-enabled applications* dapat terhubung dengan layanan *leap motion* menggunakan *leap motion* native *library* Ketika *foreground leap-enabled applications* menerima data berupa gerakan *tracking* dari layanan. Aplikasi dapat terhubung langsung dengan *leap motion* native *library* menggunakan bahasa pemrograman C++, C#, *javascript*, *labview* maupun *python*.
4. Ketika *background leap-enabled application* kehilangan koneksi dengan sistem operasi, layanan *leap motion* akan menghentikan pengiriman data yang sedang dilakukan. Aplikasi yang dimaksudkan dapat bekerja secara *background* dan dapat meminta izin untuk menerima data bahkan ketika bekerja secara *background.*

2.2.2 *Quadcopter*

Quadcopter atau *quadrotor* merupakan helikopter yang mempunyai rotor atau penggerak berjumlah 4 buah. Setiap rotor diletakkan membentuk formasi persegi dengan jarak sama dan menghadap keatas dari pusat massa *quadcopter* ditiap rotornya. Kecepatan dari *quadcopter* ditentukan dengan kecepatan sudut dari rotor yang diputar oleh motor listrik. Pada masing-masing motor terdapat baling-baling (*Propeller*) yang menghasilkan tekanan udara kebawah yang mengakibatkan adanya gaya angkat pada *quadcopter* yang menjadikan *quadcopter* bisa terbang. Jumlah baling-baling (*Propeller*) pada *quadcopter* berjumlah 4, 2 berputar searah jarum jam dan 2 lagi berputar berlawanan dengan arah jarum jam seperti pada Gambar 2.2.

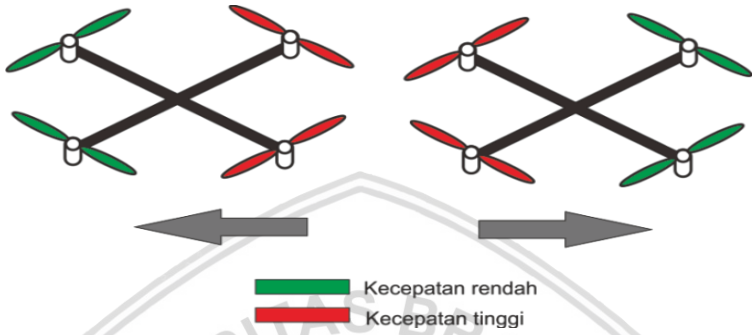


Gambar 2.2 Sistem kerja *rotor quadcopter*

Sumber : (Ardhana, 2018)

Kecepatan untuk tiap rotor dapat kita atur sesuai keinginan disetiap gerakan *roll* kanan atau bergerak kanan, *roll* kiri atau bergerak kekiri dan gerakan lainnya, kecuali untuk gerakan *takeoff*, *hover* dan mendaratnya hanya dapat diatur ketinggan untuk gerakan *takeoff* ke *hover* yang biasa diatur dengan ketinggian rata-rata 1 meter. *Quadcopter* biasanya digunakan dalam operasi penyelamatan, pemetaan wilayah, pencarian, pemantauan dan berbagai kegunaan lainnya. *Quadcopter* memiliki beberapa istilah gerakan seperti *roll*, *pitch*, *yaw* dan *gaz* untuk pengertian istilah tersebut antara lain:

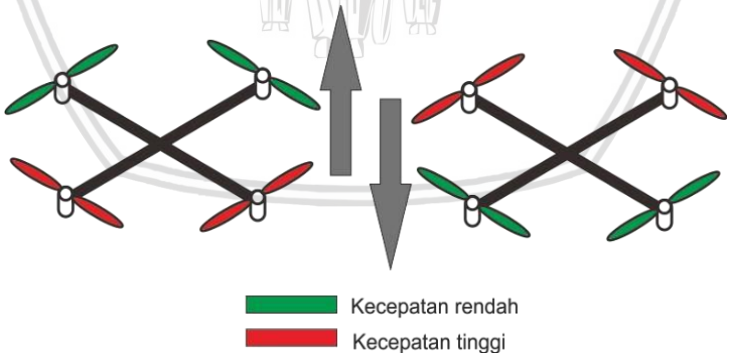
1. *Roll* merupakan gerakan pada *quadcopter* pada sumbu x dimana gerakan yang dihasilkan ialah gerakan kekanan dan kekiri seperti gambar 2.3 dibawah. Saat gerakan *roll* kanan atau gerakan ke kanan maka dua baling-baling sebelah kanan kiri kecepatannya akan ditinggikan dan dua baling-balong sebelah kanan akan direndahkan. Begitu sebaliknya ketika gerakan *roll* kiri atau gerakan ke kiri maka dua baling-baling sebelah kiri akan direndahkan kecepatannya dan dua baling-baling sebelah kanan akan ditinggikan kecepatannya.



Gambar 2.3 Gerakan *roll quadcopter*

Sumber : (Ardhana, 2018)

2. *Pitch* merupakan gerakan pada *quadcopter* pada sumbu y dimana gerakan yang dihasilkan ialah gerakan kedepan dan kebelakang seperti pada gambar 2.4 dibawah. Saat gerakan *pitch* depan atau gerakan kedepan maka dua baling-baling depan kecepatannya direndahkan dan dua baling-baling belakang kecepatannya ditinggikan. Begitu sebalikannya ketika gerakan *pitch* belakang maka bua baling-baling belakang kecepatannya direndahkan dan dua baling-baling depan kecepatannya ditinggikan.

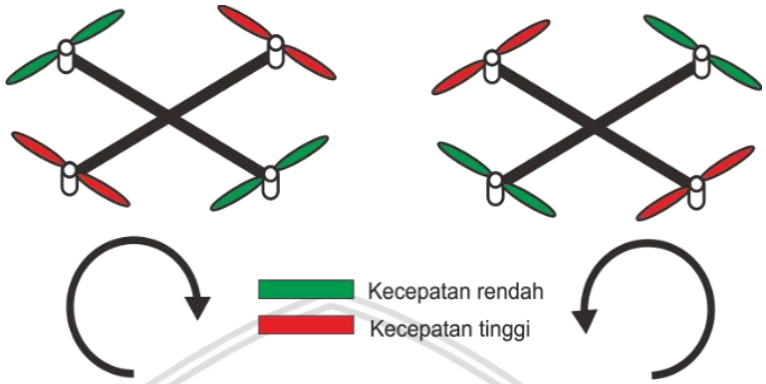


Gambar 2.4 Gerakan *pitch quadcopter*

Sumber : (Ardhana, 2018)

3. *Yaw* merupakan gerakan pada *quadcopter* pada sumbu z dimana gerakan yang dihasilkan ialah gerakan memutar kekanan dan memutar kekiri seperti pada gambar 2.5 dibawah. Perputaran baling-balingnya silang dimana saat gerakan yaw kanan atau gerakan memutar kekanan baling-baling sebelah kiri bagian belakang memutar dengan kecepatan yang tinggi dan bagian depan kecepatannya direndahakan, sebelah kanan bagian depan

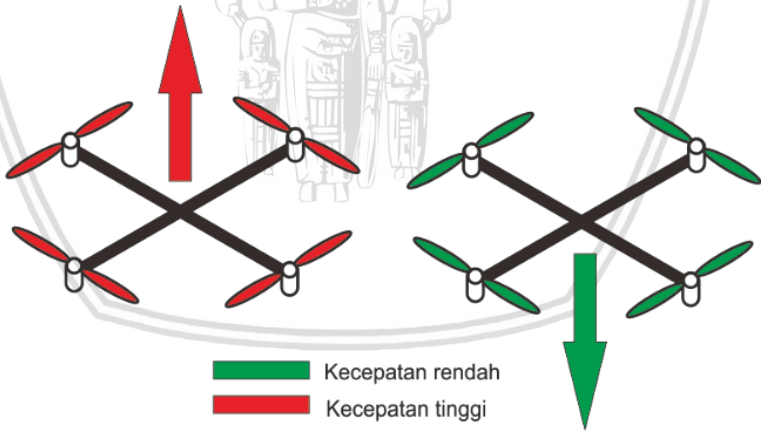
kecepatannya ditinggikan dan bagian belakan direndahkan kecepatannya. Begitu sebaliknya y*aw* kiri atau memutar ke kiri baling-baling sebelah kiri bagian depan kecepatannya ditinggikan dan bagian belakang kecepatannya direndahkan sedangkan sebelah kanan bagian depan direndahkan kecepatannya dan bagian belakang ditinggikan kecepatannya.



Gambar 2.5 Gerakan *yaw quadcopter*

Sumber : (Ardhana, 2018)

4. *Gaz* atau *Throttle* merupakan gerakan pada sumbu z dimana gerakan yang dihasilkan ialah naik dan turun seperti gambar 2.6 dibawah. Saat gerakan *gaz* naik atau gerakan naik maka keempat baling-baling kecepatannya akan ditinggikan. Begitu sebaliknya ketika gerakan *gaz* turun atau gerakan turun maka keempat baling-baling kecepatannya akan direndahkan.



Gambar 2.6 Gerakan *throttle quadcopter*

Sumber : (Ardhana, 2018)

2.2.3 *Natural User Interface*

NUI (Natural User Interface) adalah cara baru dari manusia untuk berinteraksi dengan sistem sesuai dengan apa yang ingin dilakukannya. *NUI* dirancang untuk berinteraksi secara tepat melalui sebuah konten (Ferreira, 2014). Menurut Fernández, et al., (2016) dimanfaatkan untuk pengendalian *quadcopter* dimana hal tersebut menjadi contoh cara baru dari interaksi manusia dengan komputer

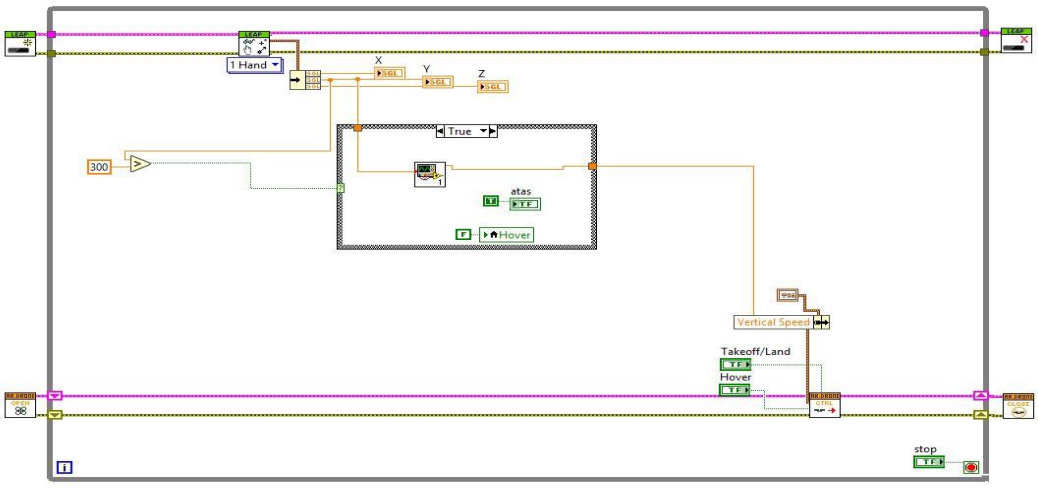
menggunakan gerakan alamiah tubuh seseorang. Terdapat beberapa kategori dalam *NUI* antara lain:

1. *Visual Body Interaction* merupakan sebuah interaksi dimana menggunakan sebuah gerakan tubuh untuk mengendalikan atau mengontrol gerakan dari *quadcopter* contohnya seperti penelitian yang dilakukan oleh Hadi et al., (2017) yaitu pengendalian *quadcopter* pemanfaatan *microsoft kinect* dimana gerakan tubuh yang dijadikan *input* kontrol gerakan.
2. *Visual Marker Interaction* merupakan sebuah interksi dimana menggunakan visual atau tanda-tanda untuk dijadikan *input* kontrol Lilian et al., (2018) yaitu sistem pendaratan otomatis *quadcopter* dengan pengolahan citra menggunakan metode *douglas puecker* dimana dalam penelitiannya menggunakan tanda-tanda seperti persegi panjang, segitiga, dan segilima sebagai *input*an kontol *quadcopter* dimana setiap kamera mendeteksi tanda tersebut *quadcopter* akan otomatis *landing* atau mendarat.
3. *Hand Gesture Interaction* merupakan interaksi dimana menggunakan gerakan tangan sebagai *input* dari gerakan *quadcopter* contohnya seperti yang dilakukan oleh Ardhana et al., (2018) yaitu mengontrol *quadcopter* menggunakan *leap motion* dimana gerakan tangan yang digunakan sebagai *input*an.
4. *Speech Command Interaction* merupakan sebuah interaksi dimana menggunakan suara sebagai kontrol dari gerakan *quadcopter* contohnya seperti yang dilakukan oleh Pallas et al., (2017) yaitu dimana menggunakan smart phone untuk mendapatkan *input*an suara sebagai kontrol dari gerakan *quadcopter* setelah *input*an didapat data suara akan dikirim menuju arduino kemudian arduino akan mengirimkan perintah gerakan kepada *quadcopter.*

2.2.4 *Labview*

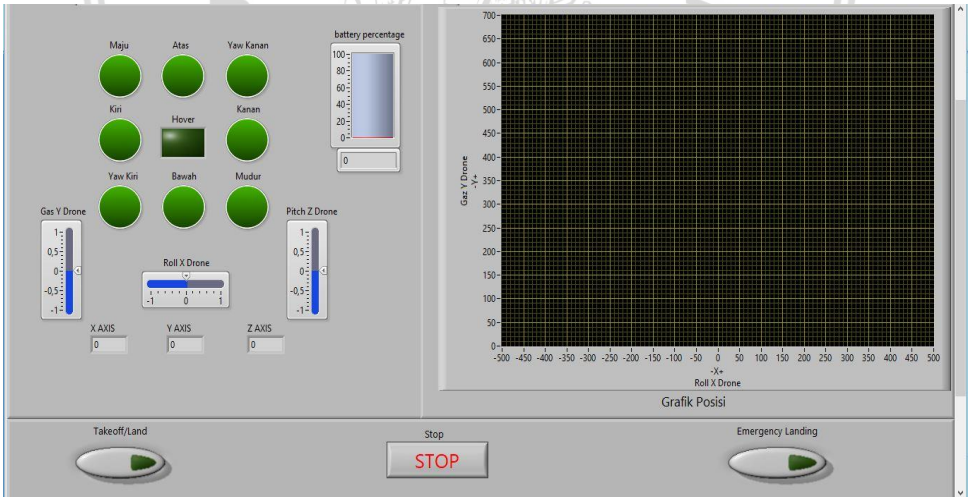
Labview adalah singkatan dari *Laboratory Virtual Instrument Engineering Workbench* merupakan sebuah platform *software* yang digunakan untuk membuat program untuk perangkat keras dan indsutri (NI: 2013). *Software* ini pertama kali dikembangkan oleh perusahaan *National Intrument* pada tahun 1986. Pemrograman yang berbasis *graphical programming* ini bertujuan untuk memudahkan pengguna ketika sedang membuat sebuah program dengan algoritma yang rumit hal tersebut mungkin dapat lebih mudah untuk dipahami, dibandingkan dengan bahasa pemrograman lain seperti C, C++ maupun *java*script yang menggunakan *textbase* dalam menulis program. Program dari *labview* disebut *VI* yaitu *Virtual Instrument* hal tersebut dikarenakan tampilan dan pengorepasiannya seperti *instrument*. Selain dari itu perusahaan *NI* juga membuat perangkat keras yang dapat langsung dihubungkan dengan bahasa pemrograman *NI Labview* sehingga sangat memudahkan *programmer* dalam memprogram perangkat keras tanpa harus memikirkan settingan komunikasi data, sinkronisasi, atau bahkan akuisisi data bahkan suatu baris. Pada *labview* terdapat beberapa fitur yaitu diagram block, *front panel, pallet tools dan pallet functoins*. Berikut penjelasan dari masing-masing fitur:

1. Diagram block merupakan tempat graphical program yang digunakan untuk program yang berupa node-node yang dihubungkan satu sama lain berisikan program-program dasar dari *labview* seperti gambar 2.7.



Gambar 2.7 Block diagram pemrograman *labview*

2. *Front panel* merupakan tampilan dari sisi luar dari program yang dapat dilihat oleh pengguna. *Front panel* sendiri berisi indikator dan kontrol yang berisikan data yang akan dikirim maunpun diterima yang akan dijadikan *input* maupun *output* oleh sistem.



Gambar 2.8 *Front panel National Instrument (NI) labview*

3. *Pallet tools* digunakan untuk memberikan tulisan, warna melilih objek dan sebagainya. *Pallet tools* sangat berhubungan dengan mouse dimana kita bisa setting secara otomatis maupun manual dengan cara tekan tombol shift + klik kanan maka *pallet* tools akan muncul seperti gambar 2.9 dimana ketika kita setting automatis maka lampu warna hijau akan menyala pada *pallet tools* dan kita bisa melakukan pekerjaan seperti menyambungkan tiap-tiap node secara otomatis pemberian tulisan secara otomatis dengan cara *double klik* dan sebagainya. Tetapi jika kita menonaktifkan otomatis pada *pallet tools* maka lampu pada *pallet tools* akan mati dan pekerjaan

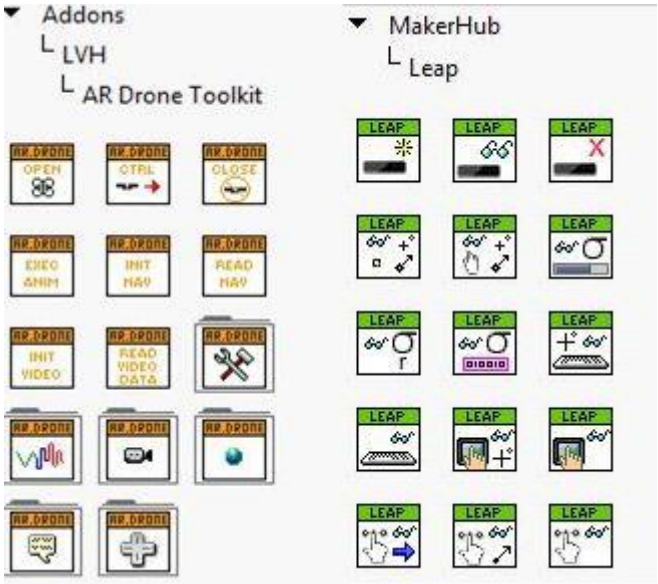
seperti memberi tulisan dan menyambungkan antar *node* akan dilakukan secara manual menggunakan fungsi-fungsi yang terdapat pada *pallet tools*.

Gambar 2.9 Gambar *Pallet Tools*

4. *Pallet functions* digunakan untuk mengambil *VI* atau *virtual instrument* yang akan dijadikan rangkaian program, terserdia tombol search digunakan untuk mempermudah pencarian *VI* yang akan dibuat program seperti gambar 2.10. pallet functions hanya berada pada diagram block dimana tempat yang dijadikan untuk penulisan program.

Gambar 2.10 Gambar *Pallet Functions*

5. Library digunakan untuk menghubungkan suatu perangkat dengan software *labview* seperti pada gambar 2.11 terdat 2 library yaitu *ar drone* dan *leap motion.*



Gambar 2.11 Library *Ar Drone* dan *Leap Motion*

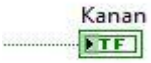
6. *Comparison* adalah contoh program yang digunakan untuk membandingkan nilai *input* dengan konstanta pada *labview* untuk menghasilkan *output* sesuai dengan perancangan contoh node program seperti pada gambar 2.12.

Gambar 2.12 *Comparison*

7. *Local Variable* adalah contoh node program yang digunakan untuk membuat variabel local yang digunakan biasanya mengakuisisi data dari variabel yang telah dibuat sebelumnya seperti gambar 2.13.

Gambar 2.13 *Local Variable*

8. Indikator merupakan node pada *labview* digunakan untuk menampilkan data contoh node program seperti gambar 2.14.



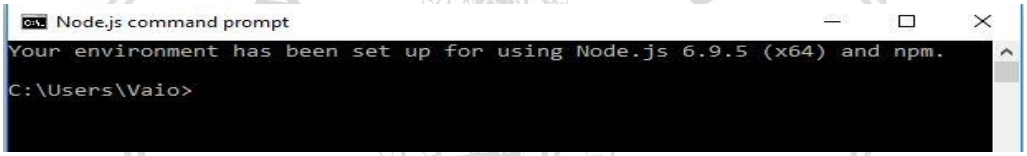
Gambar 2.14 *Indicator*

9. Kontrol pada *labview* digunakan untuk memberikan perintah perbedaan dengan indikator kontrol mengirimkan data sedangkan indikator menerima data contoh node program seperti gambar 2.15.

Gambar 2.15 Control

2.2.5 *Node js*

Node js merupakan *platform* bahasa pemrograman pengembangan dari bahasa pemrograman *javascript*. Tujuan dikembangkan *node js* supaya *javascript* tidak hanya berfokus pada pengembangan *website* saja melainkan *javascript* bisa dikembangkan untuk barbagai *platform*.



Gambar 2.16 *Node js Command prompt*

Pada Gambar 2.11 gambar *node js command prompt* merupaka tempat untuk menulis perintah pada *node js* untuk mendalikan gerakan *quadcopter*. Terdapat kumpulan-kumpulan *node* yang disebut dengan *Node Package Manager* atau yang biasa di singkat dengan nama *NPM. NPM* terintegrasi satu sama lain yang menghasilkan berbagai perintah untuk gerakan *quadcopter* seperti *takeoff*, *hover*, mendarat, ke kanan, ke kiri, dan lain lain.

2.2.6 AT Command

AT Command atau *Hayes Standartt AT Command*, merupakan sebuah standar petunjuk untuk mengendalikan atau mengonfigurasi suatu modem. *Hayes Microcomputer Products* merupakan pelopor pengembang dibidang modem yang biasa digunakan dengan *PC* (*Linux Information Project*, 2005).

AT Command adalah merupakan sekumpulan perintah yang berkomunikasi menggunakan serial *port*, biasa AT *Command* digunakan pada *modem* atau *handphone* melihat *vendor handphone* atau membaca kekuatan sinyal dari *modem* atau *handphone* yang digunakan, mengirim dan memperbarui *SMS*, dan sebagainya. *AT Command* pada dasarnya miriip dengan perintah-perintah dasar *DOS (Disk Opertaing System)* yaitu perintah yang digunakan untuk komunikasi dengan komputer melalui *port*. (Sabitha Wildani Hadi, 2017).

Format penulisan pada *AT Command* dimulai dari kata *AT* yang merupakan singkatan dari *attention*, kemudian diikuti dengan karakter lainnya yang memiliki fungsi masing-masing. Selain untuk penulisan perintah di *port*, *AT Command* juga digunakan untuk penulisan intruksi pada *modem*. Berikut contoh penulisan *AT Command:*

1. *AT* berfungsi untuk memulai penulisan *command*.
2. *AT+CGMI* digunakan untuk mengetahui merek dari *handphone.*
3. *AT+CMGR* digunakan untuk membaca *SMS*.

2.2.7 Node Package Manager

Node package manager merupakan sekumpulan perintah yang ada pada *node js* yang berfungsi untuk mempermudah programmer *javascript* dalam membuat suatu sistem dengan berbagai *code* yang tersedia, kemudian *code* tersebut dapat membantu untuk menyelesaikan penelitian. *NPM* yang digunakan penelitian kali ini ialah *Node AR Drone* 2.0 dimana *NPM* tersebut berfungsi sebagai pengatur kontrol di *quadcopter*.

2.2.7.1 *Node AR Drone 2.0*

Node AR Drone 2.0 adalah sekumpulan perintah yang ada pada *quadcopter* yang bertujuan sebagai kontol dari *quadcopter*. *Berikut contoh-contoh perintah yang ada NPM quadcopter:*

1. *Takeoff* dimana digunakan untuk mengirimkan perintah terbang pada *quadcopter*.
2. *Hover* dimana perintah tersebut digunakan untuk *quadcopter* berada pada posisi stabil ketika selesai *takeoff* dan kondisi sedang terbang.
3. *Landing* dimana perintah digunakan untuk mendarat *quadcopter*.
4. *Emergency Landing* dimana perintah digunakan untuk mendarat secara otomatis ketika *quadcopter* mengalami masalah seperti baterai *quadcopter* yang habis, *quadcopter* yang terbentur benda dan baling-baling sempat berhenti dan sebagainya.
5. *Roll* dimana perintah berfungsi untuk gerakan *quadcopter* pada sumbu x *quadcopter* contohnya gerakan ke kanan dan ke kiri.
6. *Pitch* dimana perintah berfungsi untuk gerakan *quadcopter* pada sumbu y *quadcopter* contohnya gerakan ke depan dan ke belakang.
7. *Gaz* dimana perintah berfungsi untuk gerakan *quadcopter* pada sumbu z *quadcopter* contohnya gerakan naik dan turun.
8. *Yaw* dimana perintah berfungsi untuk gerakan *quadcopter* pada sumbu z *quadcopter* contohnya meskipun sama bergerak pada sumbu z tetapi gerakan yaw digunakan untuk gerakan memutar ke kanan ataupun ke kiri itu yang menjadi perbedaan antara gerak *gaz* dan yaw.

2.2.8 Kecepatan *Quadcopter*

Kecepatan dari *quadcopter* sendiri benilai antara 0 sampai 1. Dimana 0 berarti kecepatan terendah dari *quadcopter* sendiri, sedangkan 1 adalah

kecepatan maksimal dari *quadcopter*. Kecepatan pada *quadcopter* akan diproses pada fungsi yang berada pada *AT PCMD*. Kecapatan *quadcopter* untuk penelitian kali ini didapatkan dari nilai pembacaan koordinat *leap motion* seperti persaman 2.1 dibawah:

$$V = \frac{POSISI\ LEAP\ SAAT\ DETEKSI}{\text{MAX POSISI LEAP}} \quad (2.1)$$

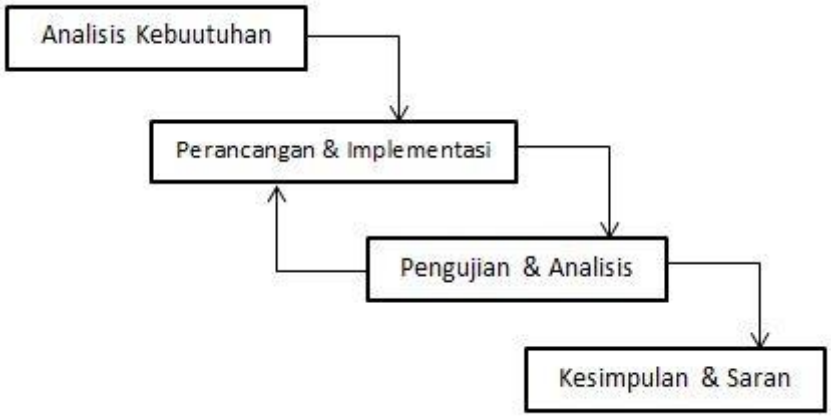
Sumber : (Ardhana, 2018)

Persamaan diatas didapat dari pengujian penentuan koordinat nilai pembacaan *leap motion* yang akan dijadikan *input* pergerakan *quadcopter*.

BAB 3 METODOLOGI

3.1 Metode Penelitian

Pada bab ini akan dijelaskan metode penelitian yang akan digunakan untuk menjalankan penelitian maupun ketika penulis hasil dari penelitian. Berikut merupakan runtutan metode yang digunakan dalam menjalankan penelitian dimulai dari analisis kebutuhan, perancangan & implementasi kemudian dilakukan pengujian serta analisis jika dalam melakukan pengujian dan analisis menemui kekurangan atau tidak sesuai dengan apa yang diharapkan maka lihat kembali perancang, baru kemudian dilanjutkan lagi prosesnya sampai akhirnya didapatkan kesimpulan dan saran seperti gambar 3.1 dibawah.



Gambar 3.1 Alur Metodologi

3.2 Analisis Kebutuhan

Pada bab ini akan dibahas kebutuhan apa saja yang diperlukan dalam penelitian contohnya seperti kebutuhan yang diperlukan oleh pengguna, kebutuhan untuk sistem, kebutuhan fungsional dan kebutuhan non fungsionalnya. Kebutuhan pengguna akan membahas apa saja yang dibutuhkan pengguna untuk berinteraksi dengan sistem. Kebutuhan sistem akan membahas tentang *hardware* dan *software* apa saja yang digunakan atau dibutuhkan oleh sistem. Kebutuhan fungsional akan membahas hubungan antara *input* dari sistem dengan pengguna tentang sistem yang diharapkan. Kebutuhan non fungsional akan membahas bagaimana karakteristik penggunanya, lingkungan operasinya, asumi dan ketergantungan pada sistem serta batasan perancangan dari perancangan sistem.

3.3 Perancangan dan Implementasi

Pada bab ini akan membahas dari perancangan sistemnya akan dirancang seperti apa dan implementasinya bagaimana. Perancangan sistem dilakukan supaya penelitian yang dilakukan akan berjalan secara sistematis tahu bagaimana gambaran dari rancangan sistem yang akan dibuat. Rancangan sistem

akan berjalan setelah analisis dari kebutuhan terpenuhi. Perancangan sistem disini akan menjelelaskan bagaimana komunikasi antara sistem dengan pengguna, gerakan tangan yang digunakan sebagai *input*, serta bagaimana perancangan kecepatan secara otomatis pada *quadcopter*. Penjelasan lebih rinci dari keterangan diatas terdapat pada bab 5 perancangan dan implementasi.

Pada implementasi sistem ini merupakan penerapan dari apa yang sudah dirancangkan oleh peneliti. Pertama yaitu perancangan komunikasi dengan sistemnya pada tahan ini akan dibuat program dimana program akan saling terhubung antara pengguna dengan sistem yang meliputi program itu sendiri, *quadcopter*, *leap motion*, serta komputer atau laptop. Kedua dilakukan implementasi dari gerakan tangan dimana pembuatan program berdasarkan koordinat yang telah disampling sebelumnya. Ketiga dilakukan implementasi dari pengiriman data dari hasil olahan *input* yang didapat menuju *quadcopter* untuk dijadikan *output*. Keempat yaitu implementasi pembuatan program untuk kecepatan otomatis *quadcopter* yang diperoleh dari hasil *input*an koordinat *leap motion*.

3.4 Pengujian dan Analisis

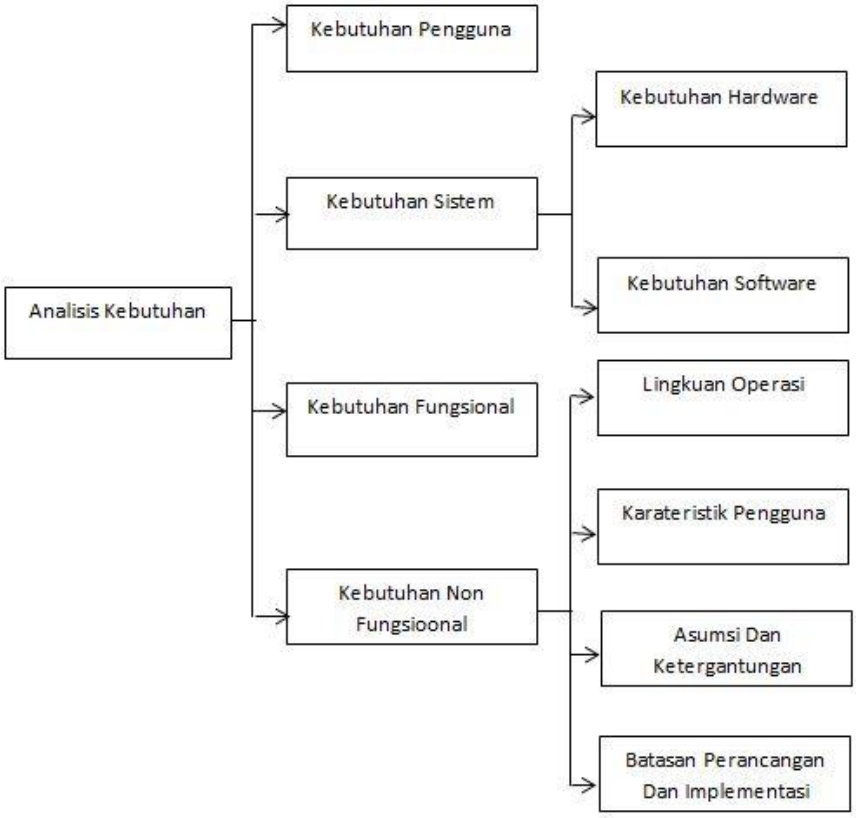
Pada bab ini akan membahas skenario pengujian yang bertujuan apakah yang telah dirancang oleh peneliti berjalan sesuai dengan yang diharapkan atau tidak. Pengujian yang akan dilakukan antara lain pengujian penentuan koordinat gerakan tangan, kecepatan *quadcopter*, ketepatan gerakan dari *quadcopter* serta *delay response* dari sistem.

3.5 Penutup

Pada bab ini terdapat kesimpulan dan saran. Dimana kesimpulan dan saran dapat diambil setelah melakukan pengujian dari apa yang sudah dirancang. Dari hasil penarikan kesimpulan dan saran penelitian akan mengetahui kelemahan dan kelebihan dari apa yang sudah diteliti. Hal ini juga dijadikan sebagai acuan atau pertimbangan untuk pengembangan penelitian selanjutnya.

BAB 4 ANALISIS KEBUTUHAN

Pada bab ini akan dibahas untuk lebih lanjut tentang analisis kebutuhan. Analisi kebutuhan akan dibagi menjadi beberapa macam yaitu kebutuhan pengguna, sistem, fungsional dan non fungsional. Dimana dari keempat kebutuhan itu masih dibagi lagi berdasarkan jenis seperti pada kebutuhan sistem yang terdapat dua yaitu kebutuhan *hardware* dan *software*. Sedangkan pada kebutuhan non fungsional dibagi menjadi empat seperti lingkungan operasi sistem, karakteristik dari pengguna, asumsi dan ketergantungan sistem serta batasan perancangan dan implementasi dari sistem seperti pada gambar 4.1 dibawah.



Gambar 4.1 Diagram analisis kebutuhan

4.1 Kebutuhan Pengguna

Kebutuhan pengguna merupakan gambaran tentang hal-hal apa saja yang dibutuhkan pengguna supaya dapat berinteraksi dengan sistem dan sistem dapat berjalan sesuai dengan keinginan pengguna. Dalam kebutuhan pengguna pada sistem ini menggunakan *front panel* pada *labview* dan *command prompt* di *Node.js* dalam menampilkan informasi yang sudah diolah oleh sistem. Informasi dari *quadcopter* yang ditampilkan meliputi:

1. Pengguna dapat mengetahui kondisi dari *quadcopter* melalui data navigasi dari *quadcopter*, data navigasi *quadcopter* berupa ketinggian dari *quadcopter*, kecepatan serta jumlah daya baterai yang ada pada *quadcopter*.
2. Pengguna dapat mengetahui poisisi tangan dari koordinat yang dihasilkan dari rekaman data posisi tangan yang ada pada *leap motion*.
3. Pengguna dapat mengetahui arah pergerakan *quadcopter* yang sedang dilakukan melalui tampilan front panel pada *labview* & *command* prompt pada *node js*.

4.2 Kebutuhan Sistem

Pada bab ini dijelaskan apa saja yang dibutuhkan oleh sistem untuk bisa berjalan sesuai keinginan pengguna. Pada bab ini kebutuhan dari sistem akan dibagi menjadi dua yaitu kebutuhan hadware dan kebutuhan dari sisi *software*.

4.2.1 Kebutuhan *Hardware*

Dalam kebutuhan *hardware* pada sistem dibutuhkan berupa *computer* atau *laptop, leap motion* serta *quadcopter* dengan jenis *Parrot AR Drone 2.0.*

4.2.1.1 *Parrot AR Drone 2.0*

Quadcpoter jenis ini pertama kali dikembangkan oleh French Company Parrot. *Parrot ar-drone 2.0* memiliki daya jelajah sebesar 50 meter, dengan dilengkapi kamera 720P yang dapat mengambil gambar dengan kualitas *HD* atau *High Definition* hasil dari pengambilan gambar maupun video juga bisa langsung diibagikan kesosial media seperti *youtube*. *Parrot ar-drone 2.0* dilengkapi dengan battery berjenis *Lithium-ion polymer* dengan *voltage* 11.1 *volt* dan daya arus sebesar 1000 mAh yang dapat dioperasikan atau digunakan untuk terbang selama kurang lebih 15 menit. *Parrot ar drone 2.0* juga dikembangkan untuk bisa dikontrol melalui sistem operasi *Windows*, *android* maupun *iOS* dengan alat kontrol berupa smartphone atau pc.

Gambar 4.2 *Parrot AR Drone* 2.0

Parrot ar drone 2.0 menyediakan sistem yang memudahkan pengguna untuk mengoperasikannya. *Quadcopter* jenis ini juga di lengkapi denga sensor akurat yang dapat bertahan ketika terjadi getaran pada *quadcopter*. Pada *quadcopter* jenis ini juga terdapat *Hull* (pelindung) yang berfungsi sebagai pengamanan

dari *quadcopter* ketika terjadi benturan seperi yang ada pada gambar 4.2 diatas. Spesifikasi *Parrot AR Drone 2.0* dapat dilihat pada Tabel 4.1 dibawah.

Tabel 4.1 Spesifikasi *Parrot AR Drone 2.0*

No	Parameter	Spesifikasi
1.	Dimensi	Badan *outdoor* memiliki dimensi : 32x 25 x 11 cm serta berat 380 gr. Sedangkan Badan *indoor* miliki dimensi : 52 x 52 x 11 cm serta berat 420 gr.
2.	Kamera	Miliki 720p dengan kualitas HD, lebar sudut lensa 92 derajat, digunakan untuk pengambilan gambar JPEG serta 30 fps untuk pengambilan video
3.	*Processor*	*ARM Cortex 1GHz 32 bit, 1 GB RAM*
4.	Sistem Operasi	*Linux 2.6.32, USB 2.0 high speed*
5.	Baterai	*Lithium polymer (1.100 mAh / 11,1 volt)*
6.	Aplikasi	*AR Freeflight 2.0 (Android)*
7.	Konektivitas	*Wifi* menngunakan *UDP port* 5555 yang digunakan sebagai pengirim sekaligus penerima perintah untuk kontrol gerakan *quadcopter*
8.	Durasi Baterai	15 menit
9.	Sup*port*	Sensor *Accelerator* 3 sumbu dengan akurasi +/- 50 mg, Sensor tekanan dengan akurasi 10 Pascal (800 cm diatas permukaan laut), Sensor *gyroscope* 3 sumbu dengan akurasi 2000 / detik, Sensor Ultrasonik untuk pengukuran tinggi, 4 *inrunner* motor *brushless* dengan, 14.5 watt daya dan kecepatan 5000 rpm

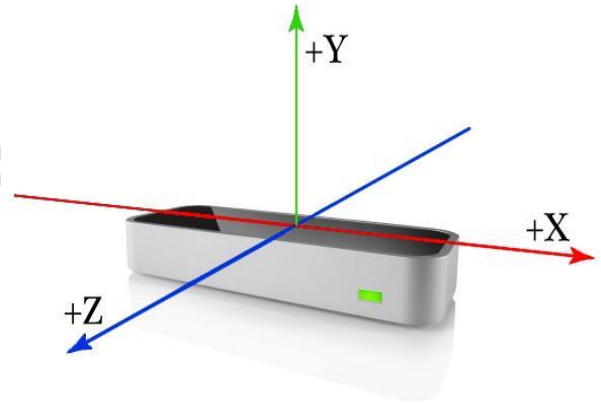
Berdasarkan spesifikasi diatas *Parrot ar drone* 2.0 pilih untuk digunakan dalam penelitian dikarena sifat *open source* yang ada pada *parrot ar drone 2.0* .

Dibandingkan dengan drone-drone yang berada di pasaran seperti *DJI* dimana program kontrolnya tidak bisa dimodifikasi sesuai dengan keinginan pengguna.

4.2.1.2 *Leap motion*

Leap motion Controller merupakan sebuah device yang menggunakan cahaya *LED* dan sensor kamera dimana *leap motion controller* memindai kedua tangan dan 10 jari dimana dapat digerakan dengan area kira-kira 8 kubik kaki (ft^3) atau sekitar (2x2x2) dari *device*. Hasil rekaman akan diolah oleh *software* dan diterjemahkan menjadi sebuah data yang akan ditampilkan kepada pengguna. (Inc,2013).

Leap motion menggunakan sensor optik dan cahaya *inframerah* yang digunakan untuk merekam data dari tangan dan jari. Perangkat bekerja pada *proximity* dengan presisi yang tinggi dalam sebuah *frame*. Sensor mengarah keatas sepanjang sumbu y sebesar 150 derajat dalam posisi normal tidak digunakan. Jarak yang efektif untuk digunakan pendeteksian adalah 25 sampai 700 milimeter dari *device*. Hal tersebut dikarenakan jarak pada *leap motion* memiliki satuan milimeter sedangkan untuk sudut menggunakan satuan radian (Inc, 2013). Pada *Leap motion* menggunakan sistem *right-hand* coordinate atau secara *defaultnya* menggunakan tangan kanan yang digunakan untuk pengambilan data koordinat dimana terdapat sumbu x, y dan z. Dimana sumbu x dan z berbentuk *horizontal* dan sumby y berbentuk *vertical* seperti pada Gambar 4.3. Oleh karena itu *leap motion* dipilih sebagai alat dalam melakukan penelitian dikarena *leap motion* dapat mendeteksi tangan dan jari yang dapat menghasilkan koordniat untuk dijadikan *input* disamping ukuran *leap motion* yang kecil dibandingkan dengan *microsoft kinect* yang ukurannya lebih besar.



Gambar 4.3 Sumbu koordinat *leap motion*

Sumber : (Inc, 2013)

4.2.1.3 Komputer / Laptop

Komputer atau laptop pada penelitian ini digunakan untuk pembuatan program serta untuk menghubungkan antara *leap motion* dan *quadcopter* untuk spesifikasi dapat dilihat pada Tabel 4.2.

Tabel 4.2 Spesifikasi Komputer

Jenis Komponen	Spesifikasi
Operating System	*Windows*® 7/8/10
Processor	AMD Phenom ™ II atau Intel® Core™
RAM	2 GB atau lebih
USB	2.0 *port*
Wi-Fi	802.11 b/g/n

Berdasarkan spesifikasi diatas maka komputer diatas dipilih dalam penelitian dikarena sesuai dengan kebutuhan dalam penelitian. Laptop digunakan untuk menghubungkan *leap motion* serta juga bisa digunakan untuk menghungkan dengan *quadcopter*. Laptop juga digunakan untuk pengolahan data dari hasil penelitian.

4.2.2 Kebutuhan *Software*

Kebutuhan *software* dalam penelitian ini antara lain *Microsoft Windows* 10 *Pro*, *Node js* v6.9.5, *SDK leap motion* orion v3.2.0, *Labview*, *Notepad++* dan *Adobe Premiere Pro CS6.*

4.2.2.1 *Microsoft Windows 10 Pro*

Microsoft windows 10 pro merupakan operating system dari laptop yang digunakan dalam penelitian berfungsi sebagai pengolah data hasil penelitian. Tempat untuk menjalankan program seperti *node js*, *labview*, SDK *leap motion*, *nodepad++*, *adobe premier*, *microsoft word* serta *microsoft excel*.

4.2.2.2 SDK *Leap motion* Orion v3.2.0

SDK (Software Develepment Kit) leap motion Orion v3.2.0 merupakan suatu *middleware* yang berfungsi sebagai *software* penghubung antara laptop atau komputer dengan *leap motion.*

4.2.2.3 *Node js v6.9.5*

Node js merupakan bahasa pemrograman bagian dari platform *javascript* yang popular dikalangan pengembang *Internet of Things* atau lebih dikenal dengan sebutan *IOT* maupun *website*. *Node js* adalah pemrograman yang berjalan disisi *server* tetapi juga bisa berjalan disisi *client*. Keutungan dari *Node js* bersifat *Asyncronous programming* dimana artinya program akan dieksekusi tanpa menunggu *state* lain selesai. *Node js* pada penelitian kali ini digunakan untuk menerima maupun mengirim data yang diperoleh dari *leap motion* menuju *mikrocontroller* yang berada pada *quadcopter* untuk dijadikan *input* gerakan pada *quadcopter* menggunakan *wifi (Wireless Fidelity)* (Foundation, 2017).

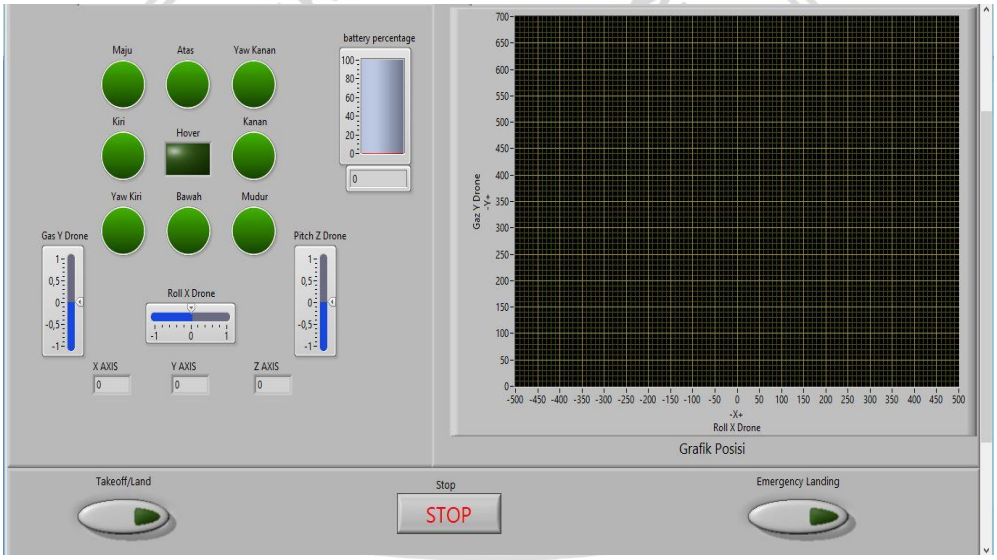
4.2.2.4 *Notepad++*

Notepad++ merupakan sebuah *text* yang berjalan pada sistem operasi *Windows*. *Notepad++* berfungsi sebagai penyunting serta menampilkan program dari berbagai bahasa pemrogramana termasuk *node js* yang digunakan sebagai bahasa pemrograman untuk mengontrol gerakan *quadcopter* pada penelitian ini.

4.2.2.5 *Labview*

Labview merupakan bahasa pemrograman yang pembuatan programnya menggunakan *National Instrument (NI) labview*, bahasa pemorograman ini biasanya digunakan untuk pembuatan program perangkat keras di industri-industri (NI: 2013). Pemrograman ini menggunakan *graphical programming* sehingga dapat memudahkan pengguna untuk mengerjakan algoritma yang rumit, karena tidak seperti bahasa pemrograman yang lain yang berbasis text.

Labview juga menyediakan GUI dimana terdapat fungsi seperti kontrol dan indikator. Fungsi kontrol berfungsi untuk membrikan perintah kontrol pada *quadcopter* sedangkan fungsi indikator berfungsi untuk menampilkan data berupa data *input* dan data *output* seperti pada gambar 2.8 dibawah ini.



Gambar 4.4 Desain Sistem Kontrol *Quadcopter* Menggunakan *Leap motion*

4.3 Kebutuhan Fungsional

Kebutuhan fungsional berfungsi untuk memberikan rincian kebutuhan apa saja yang dibutuhkan sistem supaya bisa berjalan sesuai keinginan pengguna, berikut kebutuhan dari sistem antara lain:

1. Sistem dapat menampilkan koordinat gerakan tangan serta tampilan berupa perintahan yang menjadi *input* dalam *quadcopter*. Contohnya sistem dapat menampilkan *input*an gerakan tangan seperti tangan bergeser kekiri maka koordinat yang ditampilkan berupa koordinat kiri serta tampilan berupa perintah gerakan kekiri yang akan ditampilkan di *labview* maupun *node js*.

2. Sistem dapat mengatur kecepatan serta arah dari *quadcopter*. Contohnya jika tangan pengguna bergerak kekiri sebesar 150 mm maka *quadcopter* akan bergerak kekiri dengan kecepatan tertentu sesuai *output* dari rumus kecepatan. Serta kecepatan akan semakin bertambah apabila pengguna menggeserkan tangan semakin jauh misalkan dari jarak awal 150 mm bergeser ke 320 mm maka kecepatan *quadcopter* akan semakin bertambah.
3. Sistem dapat menampilkan data Navigasi seperti persentase baterai, kecepatan dari *quadcopter*, ketinggian, *port* dan lain-lain.

4.4 Kebutuhan Non Fungsional

Kebutuhan non fungsional yang berfungsi sebagai pendukung supaya sistem dapat berjalan sesuai dengan yang diharapkan. Kebutuhan non fungsional dibagi menjadi empat yaitu karateristik dari pengguna, lingkungan operasi dari sistem, asumsi dan ketergantungan dari sistem serta batasan perancangan dan implementasi sistem.

4.4.1 Karakteristik Pengguna

Karakteristik pengguna ini bertujuan supaya pengguna pemula yang sebelumnya belum pernah mengoperasikan *quadcopter* dapat mencoba mengoperasikan *quadcopter* dengan cara mudah.

4.4.2 Lingkungan Operasi

Pada penelitian ini lingkungan operasi untuk memiliki persyaratan untuk menjalankan sistem ini adalah sebagai berikut.

1. Posisi tangan diatas leap diatur sesuai dengan kenyamanan dari pengguna.
2. pengoperasian *quadcopter* diperlukan ruangan seluas minimal 15 meter persegi agar *quadcopter* bisa terbang dengan kondisi maksimal.

4.4.3 Asumsi dan Ketergantungan

Berikut merupakan asumsi dan ketergantungan yang ada dalam sistem pada penelitian ini:

1. Sistem akan berjalan ketika komputer telah terhubung denagn *quadcopter* melalui *wifi* yang dipancarkan *quadcopter* serta *leap motion* yang juga terhung dengan komputer menggunakan *usb.*
2. Sistem akan berjalan hanya pada komputer yang telah terinstal *SDK leap motion*, *node js* serta *labview*.
3. Sistem akan berjalan ketika *leap motion* mendeteksi pergerakan tangan dan akan mengirimkan perintah berdasarkan koordniat yang dihasilkan. Ketika tangan tidak terdeteksi pada *leap motion* maka sistem akan otomatis mengirimkan perintah *landing* atau mendarat.
4. Gerakan pada *quadcopter* seperti gerakan ke depan, belakang, kanan, kiri, atas, bawah akan dijalankan setelah sistem menjalankan perintah *takeoff*.

4.4.4 Batasan Perancangan dan Implementasi

Pada bab ini berisikan batasan dari perancangan dan implementasi yang bertujuan untuk supaya sistem dapat berjalan terarah dan sesuai dengan yang diharapkan peneliti:

1. *Input* gerakan tangan hanya berfokus menggunakan tangan kanan.
2. Posisi jari harus sejajar.
3. Pada saat sistem dijalankan tidak boleh ada 2 tangan atau lebih pada area deteksi *leap motion*.
4. Area deteksi dari *leap motion* menyerupai piramid terbalik dimana batas bawah 25 mm dan batas atas 700 mm.
5. Gerakan pada *quadcopter* yang dijalankan yaitu gerakan *takeoff*, *hover*, kanan, kiri, naik, turun, depan, belakang, mendarat.
6. Setiap gerakan harus melalui gerakan *hover.*
7. Tangan harus selalu terdeteksi *leap motion* untuk menjalankan sistem.
8. Ketika tangan tidak terdeteksi *leap motion* maka *quadcopter* otomatis mendarat.

BAB 5 PERANCANGAN DAN IMPLEMENTASI

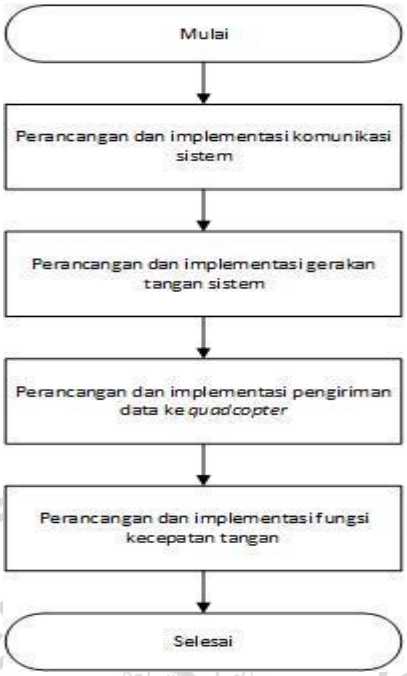
Sistem ini dibuat. Seperti yang terlihat dari gambar 5.1 pertama *leap motion* akan mendeteksi adanya gerakan tangan atau tidak. Kedua jika terdapat gerakan tangan maka data koordinat yang dihasilkan dari rekaman akan dikirimkan ke komputer untuk dijadikan *input* dari gerakan. Ketiga data hasil olahan akan dikirimkan menuju *quadcopter* untuk diajadikan *output* gerakan.

Gambar 5.1 Diagram Blok Gambaran Umum Sistem

Pada gambar 5.1 diatas terlihat mulai dari perintah pertama yaitu pengambilan data berupa koordinat yang dihasilkan dari gerakan posisi tangan yang ada diatas *leap motion*. Kemudian *leap motion* akan merekam gerakan tangan menggunakan sensor kamera dan inframerah dimana hasil remakan data berupa koordinat x, y dan z. Data akan dikirim menuju komputer untuk diolah menjadi *input* seperti pada gambar.

Dalam penelitian ini terdapat beberapa tahapan yang akan dilakukan seperti yang dapat dilihat pada gambar 5.2 yang bertujuan supaya penelitian terarah dan sesuai dengan yang diharapkan peneliti. Hal Pertama yang akan yaitu dilakukan yaitu perancangan dan implementasi komunikasi pada sistem. Kemudian dilakukan perancangan dan implementasi gerakan tangan pengguna yang nantinya kan dijadikan sebagai *input* dari sistem. Setelah itu dilakukan perancangan dan implementasi pengiriman data menuju *quadcopter* untuk dijadikan *output* berupa gerakan *quadcopter.* Terakhir dilakukan perancangan fungsi kecepatan secara otomatis. Penjelasan lebih lengkap akan dibahas pada sub bab 5.1 tentang perancangan dari komunikasi sistem pada komunikasi sistem dibahas tentang komunikasi antara sistem dengan *quadcopter* dan sistem dengan *leap motion* begitu juga dengan implementasinya. Kedua peracangan dari gerakan tangan yang akan dibahas pada sub bab 5.2, pada sub bab 5.2 ini akan dibahas posisi-posisi tangan yang digunakan untuk menjalankan sistem. Ketiga yaitu perancangan tentang pengiriman data ke *quadcopter* yang akan dibahas pada sub bab 5.3 pada bab ini akan dibahas perubahan data yang dikirimkan menuju *quadcopter* sehingga *quadcopter* berjalan sesuai input. Keempat pada sub bab 5.4 yaitu pembahasan tentang fungsi kecepatan otomatis

yang akan digunakan pada sistem, perubahan nilai koordinat menjadi input kecepatan.



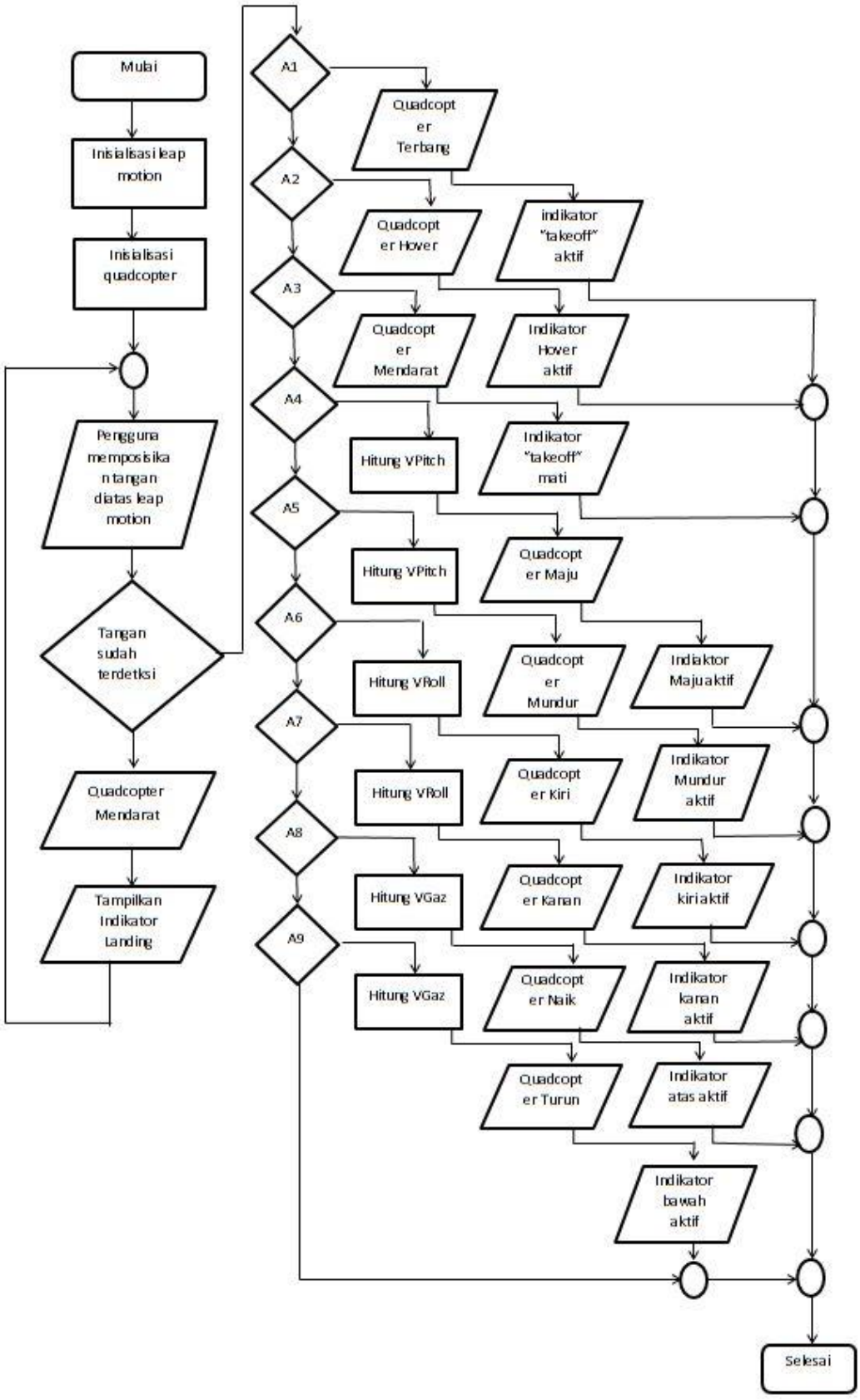
Gambar 5.2 Alur Perancangan Sistem

Flowchart untuk gambar dari sistem dapat terlihat seperti pada gambar 5.3. Pada *flowchart* terdapat istilah-istilah yang digunakan seperi A1, A2, A3 sampai dengan A9 yang akan dijelaskan lebih detail pada tabel 5.1 untuk fungsi-fungsinya. Dalam *flowchart* tahap pertama yang akan dilakukan yaitu inisialisasi atau hubungkan *leap motion* dan *quadcopter* dengan komputer. Kemudian pengguna akan memberikan *input* berupa posisi tangan diatas *leap motion* jika *leap motion* tidak mendeteksi tangan sistem akan mengirimkan perintah mendarat. Jika tangan terdeteksi maka sistem akan menjalankan perintah-perintah sesuai kode status yang ada pada tabel 5.1. Hal pertama setalah *leap motion* dan *quadcopter* terhubung dengan komputer sistem akan menjalankan setiap gerakan *quadcopter* ketika sistem telah menjalankan perintah *takeoff*. *Takeoff* memiliki kode status A1 dimana posisi gerakan tangan menggengam pada posisi *hover* di program *node js* serta posisi tangan dengan jari-jari yang melebar pada posisi *hover* di pemrogram *labview*. Pergerakan-pergerakan tersebut digunakan sebagai inisialisasi untuk memberikan *input* gerakan *takeoff* pada *quadcopter*. *Quadcopter* akan otomatis terbang setelah mendapatkan perintah *takeoff* dan sistem akan menampilkan informasi kondisi *quadcopter* sedang *takeoff* pada pada program *node js* akan tampil tulisan *takeoff* pada *command prompt* sedangkan pada *labview* lampu indikator *takeoff* akan menyala. Setelah *quadcopter takeoff* atau dalam posisi terbang sistem dapat menjalankan perintah-perintah yang telah dirancang oleh peneliti. Sebagai contoh ketika ketika pengguna menggerakkan menggerakan tangannya kekanan dari daerah *hover* maka sistem akan menjalankan kode status A7 yang artinya

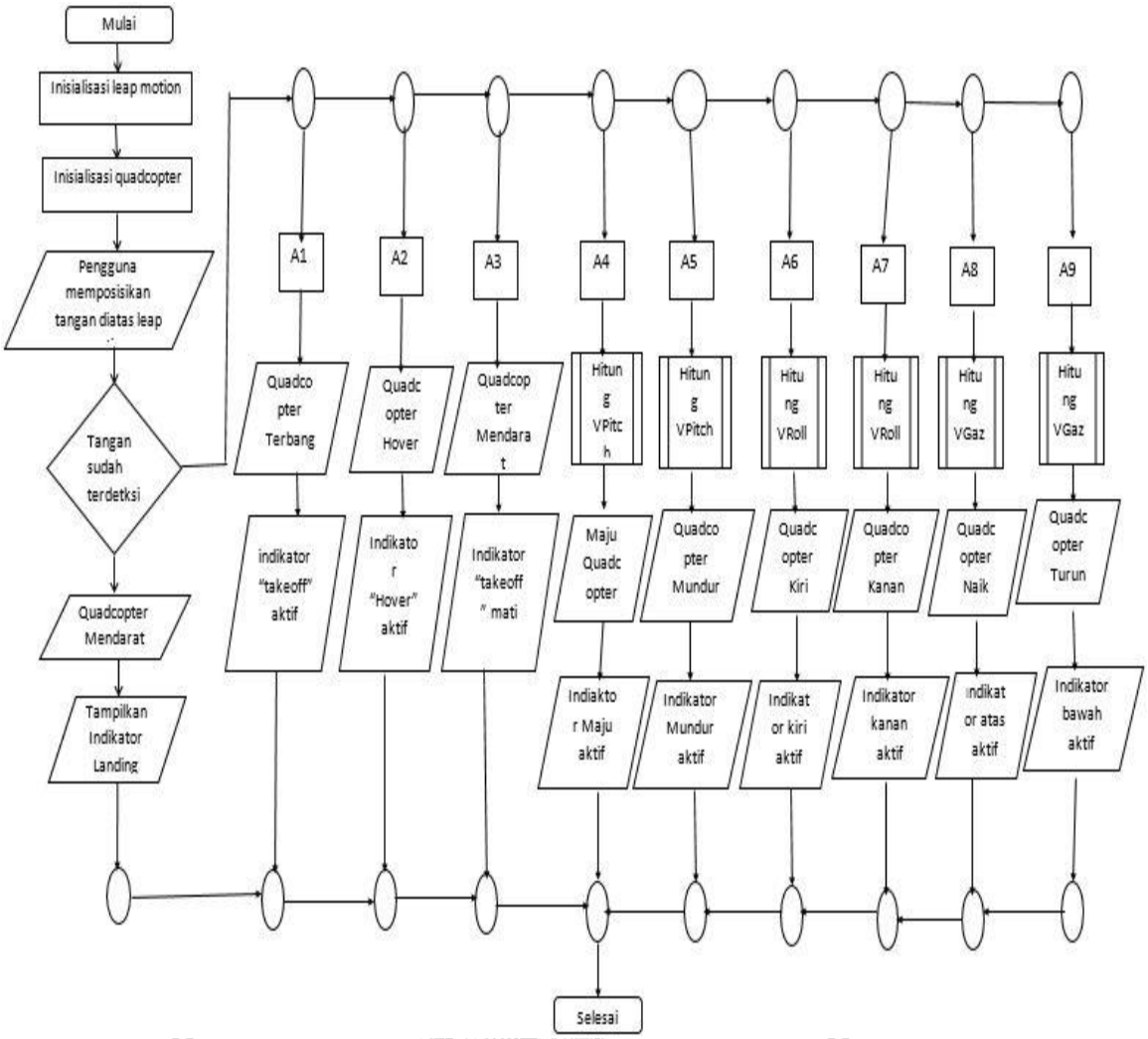
sistem akan mengirimkan perintah *roll* kekanan atau gerakan kekanan pada *quadcopter* dan sistem akan menghidupkan indikator kanan pada *labview*.

Tabel 5.1 Kode status gerakan tangan

Kode status	Gerakan Tangan Kanan	Gerakan *quadcopter*
A1	Posisi tangan dengan jari-jari yang melebar serta posisi telapak tangan yang mengggenggam pada program *node js* didaerah *hover.*	*Takeoff*
A2	Telapak tangan berada didaerah *Hover.*	*Hover*
A3	Salah satu ruas jari bergerak seperti menekan sebuah tombol pada *node js* serta membiarkan tangan tidak terdeteksi pada *leap motion* di pemrograman *labview*.	Mendarat
A4	Telapak tangan ke depan	*Pitch* ke depan
A5	Telapak tangan ke belakang	*Pitch* ke belakang
A6	Telapak tangan ke kiri	*Roll* ke kiri
A7	Telapak tangan ke kanan	*Roll* ke kanan
A8	Telapak tangan ke atas	*Gaz* ke atas
A9	Telapak tangan ke bawah	*Gaz* ke bawah



Gambar 5.3 Flowchart Sistem Kendali Navigasi *Quadcopter Node Js*



Gambar 5.4 Flowchart Sistem Kendali Navigasi *Quadcopter Labview*

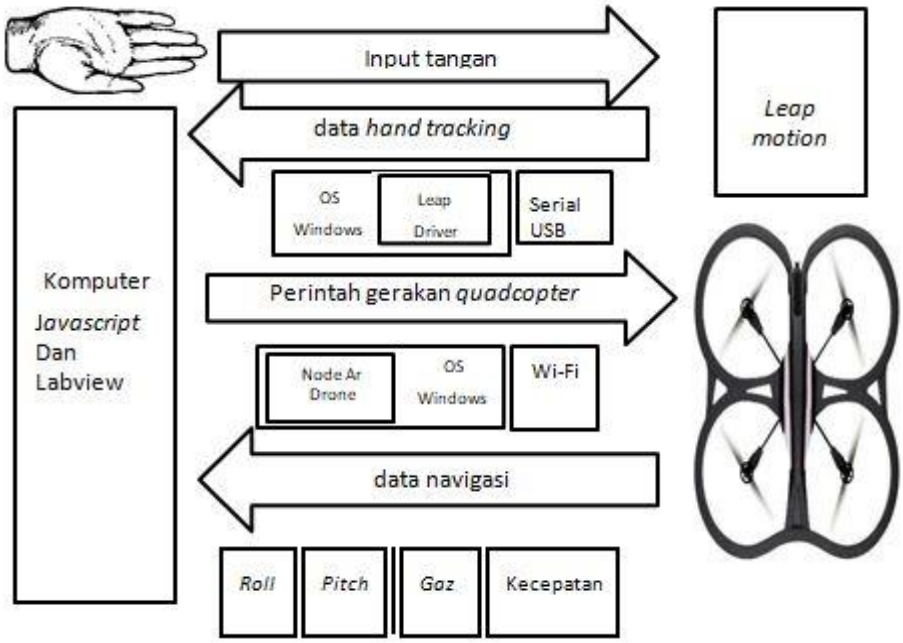
Pada gambar 5.3 dan 5.4 flowchart diatas dapat dilihat perbedaan dari kedua proses dari sistem yang digunakan pada *node js* dan *labview.* Hal tersebut tergambar dari gambar 5.3 sistem kendali *ar-drone* pada *node js* pembacaan program bersifat *sequence* atau berurutan dimulai dari baris pertama sampai baris terakhir program yang mengakibatkan proses eksekusi program memakan waktu yang lama dibandingkan proses sistem yang dapat dilihat pada gambar 5.4 sistem kendali *ar-drone* menggunakan *labview.* Dimana pada *labview* pembacaan program bersifat paralel dari kiri ke kanan pada keseluruhan program yang mengakibatkkan proses eksekusi program lebih cepat dibangdingkan *node js* yang pembacaan programnya bersifat *sequencetial* atau berurutan.

5.1 Komunikasi Sistem

5.1.1 Perancangan Komunikasi Sistem

Pada sistem ini menggambarkan tentang perancangan dari komunikasi sistem yang dapat dilihat dari gambar 5.5. Dimana sistem akan mendapatkan *input*an berupa koordinat dari gerakan tangan yang direkam oleh *leap motion*. Data akan

di olah oleh *leap motion* menggunakan *SDK leap motion* dan dikirimkan menuju komputer menggunakan serial usb 2.0. Pada komputer ini menggunakan data yang berada pada *SDK* atau *Software Development Kit* dari *leap motion* akan di olah untuk dijadikan *input* pada sistem pada pemrogram *node js* dan *labview*. Setalah hasil olahan data koordinat selesai dirubah menjadi bentuk *input*an gerakan *quadcopter*, data *input*an akan dikirim menuju *quadcopter* untuk dijadikan *output* gerakan menggunakan koneksi wifi *UDP port* 5555 dimana fungsi dari *port* digunakan untuk proses pengiriman dan penerimaan data navigasi dari *quadcopter*. Setelah perintah dijalankan *quadcopter* akan mengirimkan data navigasi terbaru menuju komputer dan data navigasi akan ditampilkan informasi berupa gerakan dari *input* yang sedang dijalankan. Sebagai contoh ketika *input* tangan bergerak ke kiri dari posisi *hover* maka *quadcopter* akan meneriman *input roll* left yang artinya*quadcopter* akan bergerak ke kiri dan sistem akan menampilkan informasi gerakan *quadcopter* jika pada *node js* berupa tulisan " *turn left the quadcopter*" berbeda dengan *labview* yang menggunakan indikator dimana indikator "kiri" akan menyala ketika *quadcopter* bergerak kekiri.

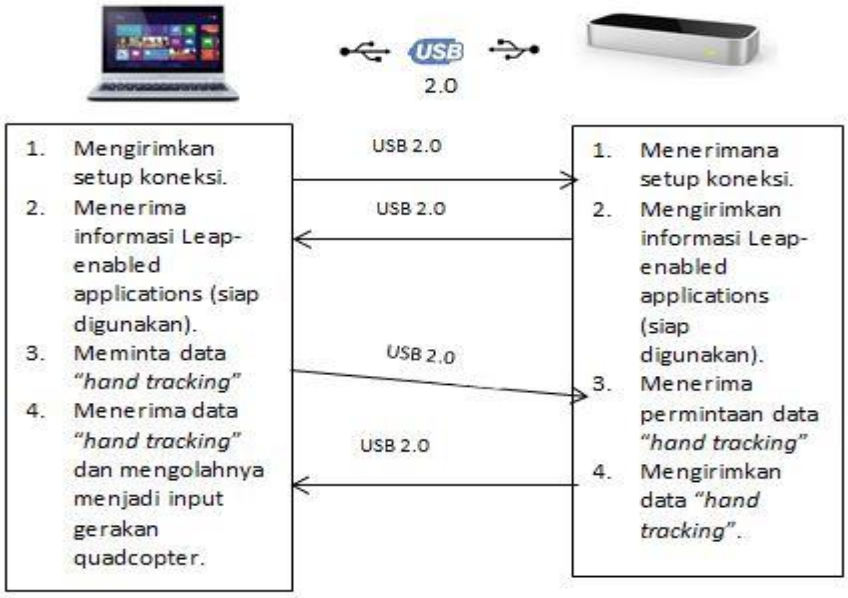


Gambar 5.5 Alur Pertukaran Data Pada Sistem

5.1.2 Implementasi Komunikasi Sistem

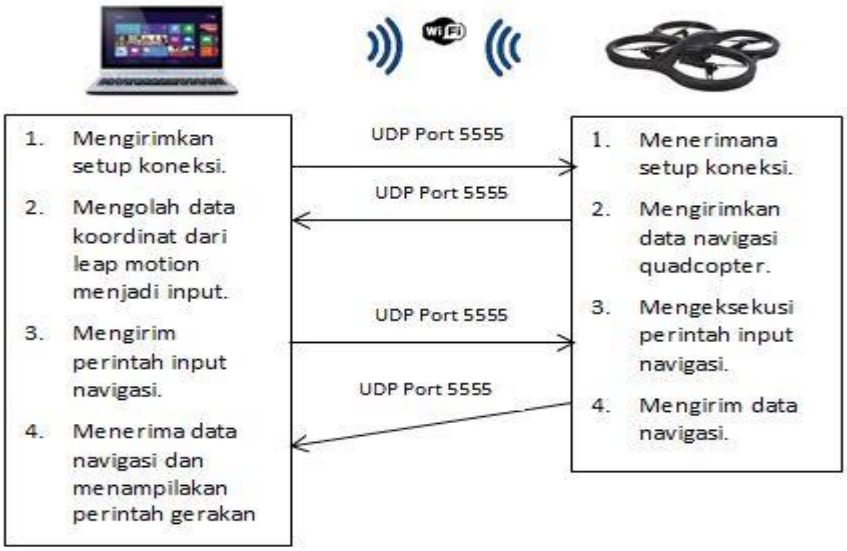
Pada bab ini akan dijelaskan tentang implementasi komunikasi yang ada pada sistem. Hal pertama yang dilakukan adalah menghubungkan semua perangkat yang digunakan *leap motion quadcopter* beserta komputer. Kemudian sistem akan dapat dijalankan ketika pengguna memberikan *input* berupa gerakan tangan yang dilakukan diatas *leap motion*. Komunikasi antara *leap motion* dan komputer dapat dilihat pada gambar 5.6. Dimana dari gambar tersebut dapat dilihat proses dimana sistem dapat terhubung dengan *leap motion* menggunakan *usb 2.0* untuk pertukaran datanya. Data yang dihasilkan

dari pergerakan tangan pengguna yang terdeteksi oleh *leap motion* berupa data *hand tracking* akan diolah pada *leap motion SDK*. Kemudian data tersebut bisa diakses menggunakan *Hand Position Velocity.vi* pada pemrograman *labview* dan class *leap-0.6.4.js* pada pemrograman *node js*. Setelah data berhasil didapatkan data akan diolah untuk dijadikan *input* gerakan pada *quadcopter*.



Gambar 5.6 Alur Komunikasi Sistem Dengan *Leap Motion*

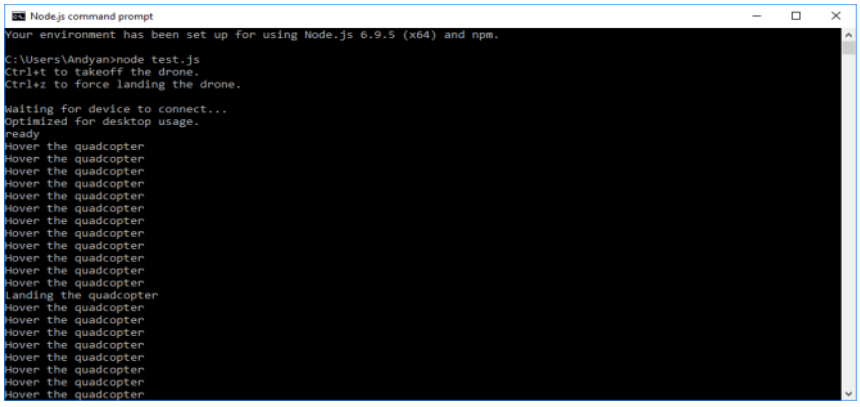
Pada program utama dibuat class seperti coba uji.vi pada pemrograman *labview* dan sistemkontrol*quadcopter*.js pada pemrograman *node.js* dimana program tersebut merupakan program utama untuk mengontrol sistem dari navigasi *quadcopter*. Class tersebut mengambil data *input* berupa *hand tracking* dari hand position *velocity*.vi pada program *labview* dan *class leap-0.6.4.js* pada program *node.js* data tersebut akan dirubah menjadi *input* berupa pergerakan *quadcopter* seperti *takeoff*, *hover*, mendarat, kanan, kiri dan pergerakan lainnya. Data *hand tracking* juga dipergunakan untuk membuat fungsi kecepatan pada *quadcopter* dimana kecepatan *quadcopter* terdiri dari 0 sampai 1 untuk penjelasan lebih lanjut tentang kecepatan akan dijelaskan pada sub bab 5.4 yang akan membahas tentang fungsi kecepatan. Sedangkan untuk komunikasi antara sistem dengan *quadcopter* dapat dilihat pada gambar 5.7 dimana pada gambar tersebut dijelaskan proses dari mulai munghubungkan *quadcopter* dengan komputer menggunakan koneksi wifi *UDP* pada *port* 5555 yang digunakan untuk pertukaran data berupa navigasi dari *quadcopter* sampai pengiriman data koordinat dari *hand tracking* yang dijadikan *input* diolah menjadi *output* berupa gerakan dari *quadcopter*. Kemudian data navigasi pergerakan juga ditampilkan pada sistem melalui *interface* yaitu *command prompt* dari pemrogram *node.js* seperti pada gambar 5.7 dan *front panel* pada pemrograman *labview* seperti pada gambar 5.8 yang berfungsi sebagai penyampai informasi terhadap pengguna.



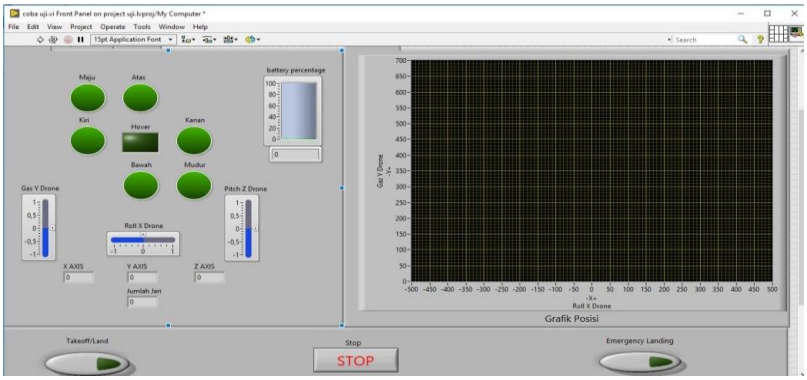
Gambar 5.7 Alur Komunikasi Sistem Dengan *Quadcopter*

Dari gambar 5.7 diatas dapat dilihat untuk proses yang terjadi pada komputer dan *quadcopter* dimana proses diawali dengan meminta koneksi untuk menghubungkan antara komputer dengan *quadcopter*. Kemudian *quadcopter* akan menerima koneksi dan akan mengirimkan data navigasi ke komputer. Setelah itu data koordinat dari *leap motion* diolah pada pemrograman *node js* dan *labview* untuk dijadikan *input* gerakan pada *quadcopter* selanjutnya perintah *input* yang sudah dikirim akan dieksekusi oleh *quadcopter* menjadi gerakan sekaligus *quadcopter* akan mengirimkan data navigasi dari *quadcopter* dan komputer akan menerima data navigasi dari *quadcopter* dan ditampilkan menjadi perintah gerakan dari *quadcopter* pada komputer.

komunikasi keseluruhan pada *quadcopter* dipusatkan pada *Control Drone.vi* pada pemrogram *labview* dan *class client.js* pada pemrograman *node.js*. Pada *Control Drone.vi* dan class *client.js* tersebut mengatur komunikasi *quadcopter* mulai dari pengiriman *AT Command* sampai penerimaan data navigasi dari *quadcopter*. Penjelasan lebih lanjutnya akan dibahas pada sub bab 5.3 yang membahas tentang pengiriman data ke *quadcopter.*



Gambar 5.8 Tampilan Sistem Menggunakan *Command Prompt* Pada *Node js*



Gambar 5.9 Tampilan Sistem Menggunakan *Front Panel* Pada *Labview*

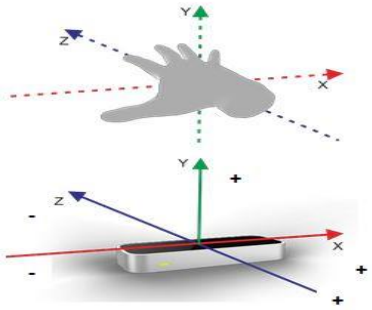
Pada gambar 5.5 merupakan gambaran sistem pada pengendalian *quadcopter*. Pada saat pengguna memposisikan tangan diatas *leap motion* dan melakukan pergerakan dari pergerakan tersebut akan diambil data *input* dan dirimkan menuju *quadcopter* untuk dijadikan *output* berupa gerakan *qudcopter*. Setelah itu perintah gerakan ditampilkan pada *interface* program seperti yang terlihat pada gambar 5.8 dan 5.9 diatas.

5.2 Gerakan Tangan

5.2.1 Perancangan Gerakan Tangan

Pada bab ini akan membahas tentang perancangan gerakan tangan yang akan dijadikan *input* dalam sistem. *Input* pada sistem didapat dari data *hand tracking* pada *leap motion.* Terdapat beberapa data *hand tracking* yang digunakan dalam sistem antara lain position yang terbagi menjadi 2 di *labview* yaitu *hand position velocity* yang digunakan untuk gerakan *roll*, *pitch*, *gaz*, *hover* dan *N point position velocity* yang digunakan untuk gerakan *takeoff* dan mendarat pada pemrograman *labview*. Sedangakan pada pemrograman *node js* data *hand tracking* yang digunakan yaitu *handheld* atau keadaan tangan menggengam untuk gerakan *takeoff, key tap* untuk gerakan mendarat dan position untuk gerakan *roll*, *pitch*, *gaz* dan *hover*. Setiap data *hand tracking* memiliki koordinat yaitu koordinat x, y dan z. Koordinat x yaitu koordinat yang ada pada sumbu x *leap motion* dimana nilai x kekanan bernilai positif dan x kekiri bernilai negatif digunakan untuk *input* gerakan *roll*. Koordinat y yaitu koordinat yang ada pada sumbu y *leap motion* dimana nilai y tangan semakin bergerak keatas maka nilai semakin besar digunakan untuk gerakan *gaz*. Koordinat z yaitu koordinat yang ada pada sumbu z *leap motion* dimana nilai z bernilai positif untuk gerakan mundur dan negati untuk gerakan maju seperti pada gambar 5.10.

Dalam bab ini dibahas tentang bagaimana penentuan koordinat yang dijadikan *input* pada sistem. Koordinat yang digunakan dihasilkan dari perobaan yang berulang-ulang. Hasil dari percobaan penetuan koordinat bisa dilihat pada bab 6.1 dan lampiran.



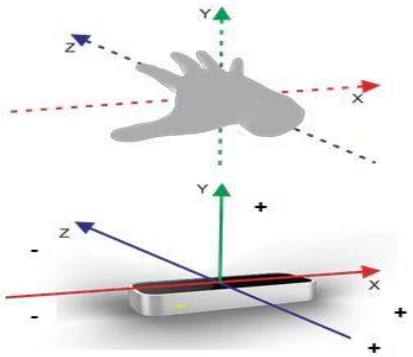
Gambar 5.10 Koordinat Tangan Dengan *Leap Motion*

Sumber : (Ardhana, 2018)

5.2.1.1 Gerakan *Takeoff*

Gerakan *takeoff* merupakan gerakan yang digunakan untuk menjalankan perintah terbang pada *quadcopter*. Perintah *takeoff* di pemrograman *node.js* dijalankan dengan cara menggenggam pada koordinat sama dengan gerakan *hover* hal tersebut dilakukan karena pada pemrograman *node.js* gerakan *takeoff* menggunakan data *hand tracking* berupa *handheld* yang artinya ketika tangan posisi menggenggam akan mengaktifkan gerakan *takeoff* seperti pada gambar 5.11. Sedangkan perintah *takeoff* pada pemrograman *labview* dijalankan dengan cara memposisikan tangan diatas *leap motion* dengan melebarkan jari-jari dikarenakan pada *labview* menggunakan data *hand tracking* berupa *N point position velocity* dimana data tersebut digunakan untuk mendeteksi jumlah jari pengguna. Batasan-batasan penentuan koordinat gerakan *takeoff* bisa dilihat pada tabel 6.7 pada pemrograman *node.js* dan tabel 6.8 pada pemrograman *labview*. Berikut batasan nilai koordinat x, y dan z yang digunakan untuk menghasilkan gerakan *takeoff* yaitu:

a. Koordinat x : posisi tangan > -60 dan posisi tangan < 30
b. Koordinat y : posisi tangan > 200 dan posisi tangan < 300
c. Koordinat z : posisi tangan > -40 dan posisi tangan < 70



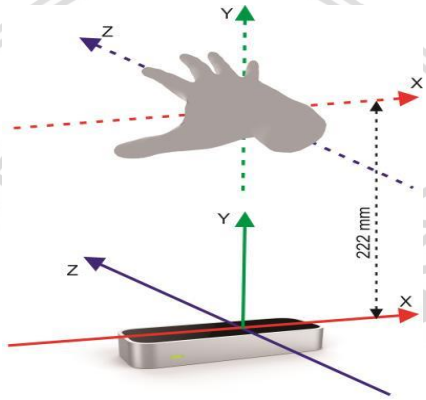
Gambar 5.11 Gerakan *Takeoff*

Sumber : (Ardhana, 2018)

5.2.1.2 Gerakan *Hover*

Gerakan *hover* yang digunakan untuk menstabilkan posisi dari *quadcopter*. Cara menjalan gerakan *hover* pada pemrograman *labview* dan *node.js* sama yaitu dilakukan dengan cara menempatkan posisi tangan diatas *leap motion* dengan jarak kurang lebih 222 mm pada sumbu y *leap motion* seperti gambar 5.12. Berikut merupakan batasan nilai koordinat x, y dan z yang diperoleh dari percobaan penentuan koordinat pada tabel 6.7 yang dilakukan pemrograman *node.js* dan pada tabel 6.8 yang dilakukan pada pemrograman *labview* yang digunakan untuk menghasilkan gerakan *hover* ini yaitu:

a. Koordinat x : posisi tangan > -60 dan posisi tangan < 30
b. Koordinat y : posisi tangan > 200 dan posisi tangan < 300
c. Koordinat z : posisi tangan > -40 dan posisi tangan < 70



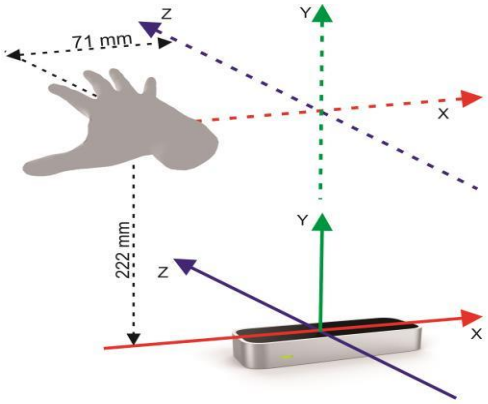
Gambar 5.12 Gerakan *Hover*

Sumber : (Ardhana, 2018)

5.2.1.3 Gerakan ke Kiri

Gerakan ini merupakan perintah yang diberikan untuk menggerakan *quadcopter* ke kiri, gerakan ini termasuk dalam gerakan *roll*. Gerakan ini dilakukan dengan cara memposisikan tangan *diatas leap motion* pada jarak kurang lebih 71 mm pada sumbu x *leap motion* seperti pada gambar 5.13. Nilai positif dan negatif pada sumbu x *leap motion* menentukan posisi tangan dan gerakan yang dihasilkan oleh *quadcopter*. Dimana nilai negatif pada sumbu x menunjukkan posisi tangan ada pada posisi kiri dan akan menghasilkan gerakan ke kiri pada *quadcopter*. Berikut nilai batasan koordinat x, y dan z yang digunakan untuk menghasilkan gerakan *roll* kiri atau gerakan ke kiri yang didapat dari percobaan pada table 6.3 dan 6.4 yaitu:

a. Koordinat x : posisi tangan < -60
b. Koordinat y : posisi tangan > 200 dan posisi tangan < 300
c. Koordinat z : posisi tangan > -40 dan posisi tangan < 70



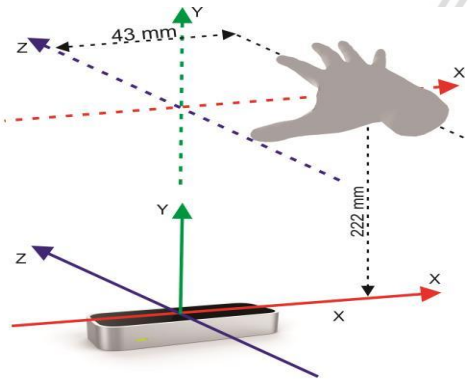
Gambar 5.13 Gerakan Ke Kiri

Sumber : (Ardhana, 2018)

5.2.1.4 Gerakan ke Kanan

Gerakan ini merupakan perintah yang diberikan untuk menggerakkan *quadcopter* kekanan, perintah ini termasuk dari gerakan *roll* pada *quadcopter*. Gerakan ini dijalankan dengan cara memposisikan tangan diatas *leap motion* dengan ketigian kurang lebih 250 mm kemudian gerakkan tangan kekanan pada sumbu x *leap motion* kurang lebih 43 mm seperti pada Gambar 5.14. Nilai postif dan negatif pada sumbu x menentukan posisi dari tangan dan *output* dari gerakan *quadcopter*. Dimana pada ini sumbu x bernilai positif yang akan mengahasilkan gerakan kekanan pada *quadcopter*. Berikut merpukan batasan dari nilai koordinat yang akan digunakan untuk menghasilkan gerakan *roll* kanan yang didapat dari percobaan pada table 6.3 dan tabel 6.4 yaitu:

a. Koordinat x : posisi tangan > 30
b. Koordinat y : posisi tangan > 200 dan posisi tangan < 300
c. Koordinat z : posisi tangan > -40 dan posisi tangan < 70



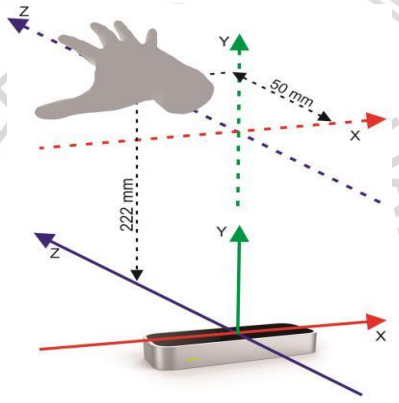
Gambar 5.14 Gerakan Ke Kanan

Sumber : (Ardhana, 2018)

5.2.1.5 Gerakan ke Depan

Gerakan ini merupakan perintah yang digunakan untuk memberi perintah bergerek ke depan atau maju, gerakan ke depan termasuk gerak *pitch* dalam *quadcopter*. Gerakan ini dijalankan dengan cara memposisikan tangan diatas *leap motion* kemudian gerakkan ke depan pada sumbu z *leap motion* kurang lebih 50 mm sepertti pada gambar 5.15. Nilai positif dan negatif menentukkan posisi tangan dimana negatif menunjukkan posisi tangan sedang berada pada posisi ke depan dalam sumbu z *leap motion*. Berikut merupakan batasan nilai koordinat yang digunakan untuk menghasilkan gerakan *pitch* depan yang didapat dari percobaan pada tabel 6.1 dan tabel 6.2 yaitu:

a. Koordinat x : posisi tangan > -60 dan posisi tangan < 30
b. Koordinat y : posisi tangan > 200 dan posisi tangan < 30
c. Koordinat z : posisi tangan < -40



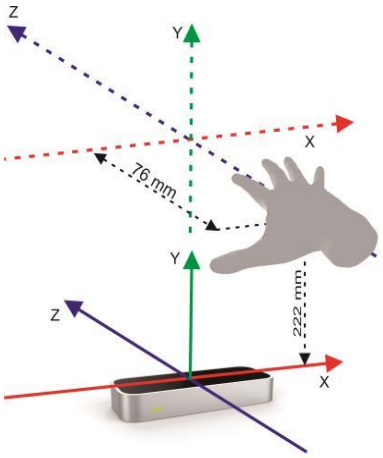
Gambar 5.15 Gerakan Ke Depan

Sumber : (Ardhana, 2018)

5.2.1.6 Gerakan ke Belakang

Gerakan ini merupakan perintah yang digunakan untuk menggerakkan *quadcopter* ke belakang, gerakan ini termasuk gerakan *picth* pada *quadcopter*. Gerakan ini dijalankan dengan cara memposisikan tangan diatas *leap motion* dengan ketinggian kurang lebih 250 mm pada sumbu y *leap motion* dan gerakkan kebelakang kurang lebih 76 mm seperti pada gambar 5.16. Berikut merupakan batasan nilai koordinat yang digunakan untuk menghasilkan gerakan *pitch* belakang yang didapatkan dari hasil percobaan pada table 6.1 dan tabel 6.2 yaitu:

a. Koordinat x : posisi tangan > -60 dan posisi tangan < 30
b. Koordinat y : posisi tangan > 200 dan posisi tangan < 300
c. Koordinat z : posisi tangan > 70



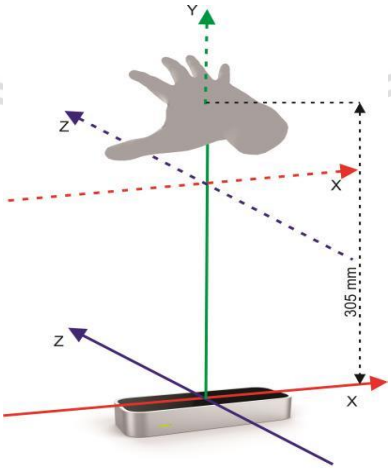
Gambar 5.16 Gerakan Ke Belakang

Sumber : (Ardhana, 2018)

5.2.1.7 Gerakan Ke Atas

Gerakan ini merupakan perintah yang digunakan untuk mejalankan gerakan ke atas pada *quadcopter*, gerakan ini termasuk gerakan *gaz*. Gerakan ini dijalankan dengan cara memposisikan tangan pada sumbu y *leap motion* pada ketinggian kurang lebih 250 mm kemudian lakukan gerakan keatas pada telapak tangan pada sumbu y kurang lebih 305 mm seperti pada gambar 5.17 . Berikut merupak batasan nilai koordinat yang digunakan untuk menghasilkan gerakan *gaz* atas yang di peroleh dari percobaan pada tabel 6.5 dan tabel 6.6 yaitu:

a. Koordinat x : posisi tangan > -60 dan posisi tangan < 30
b. Koordinat y : posisi tangan > 300
c. Koordinat z : posisi tangan > -40 dan posisi tangan < 70



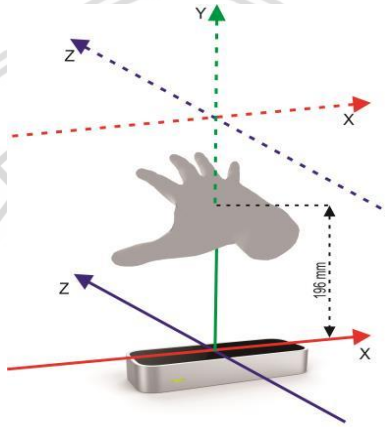
Gambar 5.17 Gerakan Ke Atas

Sumber : (Ardhana, 2018)

5.2.1.8 Gerakan ke Bawah

Gerakan ini perintah yang digunakan untuk menjalankan gerakan *quadcopter* ke bawah, gerakan ini termasuk gerakan *gaz* pada *quadcopter*. Gerakan ini dijalankan dengan cara memposisikan telapak tangan pada sumbu y *leap motion* kurang lebih 250 mm kemudian turunkan ke bawah sampai ketinggian kurang lebih 196 mm seperti pada gambar 5.18. Berikut merupakan batasan nilai koordinat yang digunakan untuk menghasilkan gerakan *gaz* bawah yang diperoleh dari percobaan pada tabel 6.5 dan tabel 6.6 yaitu:

a. Koordinat x : posisi tangan > -60 dan posisi tangan < 30
b. Koordinat y : posisi tangan < 200
c. Koordinat z : posisi tangan > -40 dan posisi tangan < 70



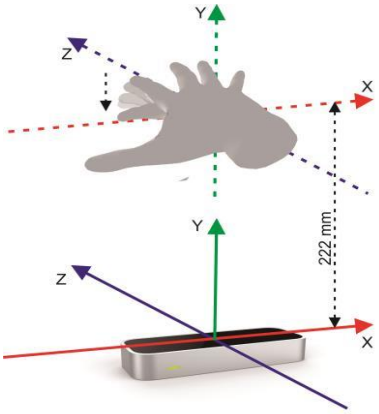
Gambar 5.18 Gerakan Ke Bawah

Sumber : (Ardhana, 2018)

5.2.1.9 Gerakan Mendarat

Gerakan ini merupakan perintah yang digunakan untuk menggerakan *quadcopter* mendarat, gerakan ini dinamakan *landing* pada *quadcopter*. Gerakan ini dijalankan dengan cara menekan pada pemrograman *node js* dan pada pemrograman *labview* cukup hanya membiarkan tidak ada tangan yang terdeteksi pada *leap motion* maka *quadcopter* akan otomatis mendarat seperti gambar 5.19. Berikut merupakan nilai koordinat yang digunakan untuk menghasilkan gerakan *landing* yang diperoleh dari percobaan pada tabel 6.7 dan 6.8 yaitu:

a. Koordinat x : posisi tangan > -60 dan posisi tangan < 30
b. Koordinat y : posisi tangan > 200 dan posisi tangan < 300
c. Koordinat z : posisi tangan > -40 dan posisi tangan < 70



Gambar 5.19 Gerakan Mendarat

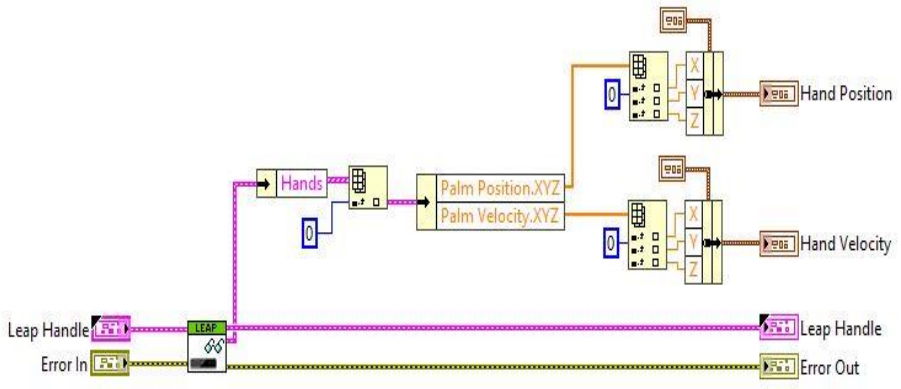
Sumber : (Ardhana, 2018)

5.2.2 Implementasi Gerakan Tangan

Pada bagian implementasi tangan hal permata yang dilakukan adalah inisialisasi varibel yang digunakan untuk mendeteksi tangan yaitu *frame* yang berisikan informasi satu set tangan dan jari yang akan dilacak datanya yang terdeteksi dalam satu *frame*, *type* bersungsi sebagai pemilihan tipe-tipe *hand tracking* yang digunakan seperti *position* berfungsi sebagai nilai koordinat dari telapak tangan. Data *hand tracking* diambil dari class *leap-0.6.4.js* pada pemrograman *node js* dan pada *labview library Read Hand Position Velocity.vi* seperti pada gambar 5.20.

Kode Program 5.1 Inisialisasi variable

```
1  controller.on('frame', function(frame) {
2  var hand = frame.hands[0];
3  var type = hand.type;
4  var YawRadians = hand.Yaw();
5  var position = hand.palmPosition;
6  });
```



Gambar 5.20 Kode Program *Read Hand Position Velocity.vi*

Gambar 5.20 diatas merupakan bagian dari *library* yang menyimpan data dari *input*an posisi tangan berupa sudut XYZ yang akan diolah sebagai *input*an pada *quadcopter* dimana sudut X akan digunakan menjadi *input*an gerakan *roll quadcopter*, Y akan digunakan sebagai *inputan vertical speed* atas *gaz* dan Z akan digunakan sebagai *input*an *pitch quadcopter*.

Setelah dilakukan inisialisasi variabel yang dibutuhkan untuk mendeteksi tangan, selanjutnya pembuatan kode program untuk masing-masing gerakan sesuai pada bab perancangan gerakan tangan.

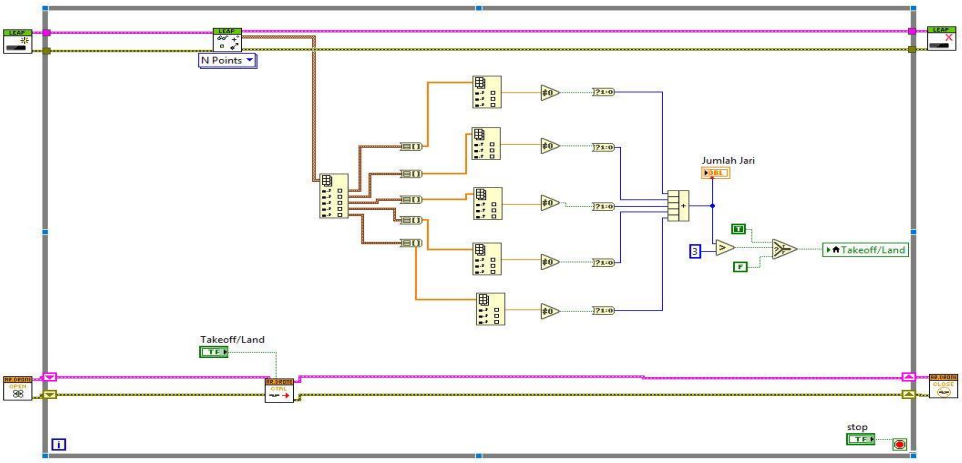
5.2.2.1 Gerakan *Takeoff*

Gerakan *takeoff* merupakan gerakan dimana *quadcopter* akan terbang. Gerakan *takeoff* pada program *node.js* menggunakan *handstatefromhistory*. Pada baris kedua mempunyai makna jika tangan menggenggam bernilai sama dengan 1. Untuk baris ketiga yaitu jika tipe tangan adalah kanan maka baris keempat yaitu mengirim perintah menuju *quadcopter* untuk terbang. Lalu untuk bari kelima yaitu menampilkan tulisan pada *command prompt* "*Takeoff* the *quadcopter*".

Kode Program 5.2 Gerakan *takeoff node js*

```
1  function handStateFromHistory(hand, historySamples){
2  if(hand.grabStrength == 1) {
3  if (type == "right"){
4  client.Takeoff();
5  console.log('Takeoff the drone.');
6  }
7  }
8  }
9  handStateFromHistory(hand, 1);
```

Sedangkan pada pemrograman *labview takeoff* menggunakan cara yang berbeda. Pada *labview* penulis menggunakan 3 titik point jari untuk menjalankan perintah *takeoff*. Jika *leap motion* mendeteksi adanya 3 titik jari maka *quadcopter* akan otomatis *takeoff*. Untuk contoh program program akan ditunjukkan pada gambar 5.21.



Gambar 5.21 Program *Takeoff* dan *Landing* Pada *Labview*

Pada contoh program gambar 5.21 merupakan program dimana akan digunakan sebagai program *takeoff* dan *landing* otomatis pada pemrograman *labview*. Pada program tersebut akan mejalankan *takeoff* jika titik titik point jari terdeteksi lebih dari 3 maka akan bernilai *true dan* otomatis akan menjalankan perintah *takeoff* pada *quadcopter*, sebalikan jika kurang dari 3 titik point jari yang terdeteksi maka program akan bernilai *false* dan mengirimkan perintah *landing* pada *quadcopter*.

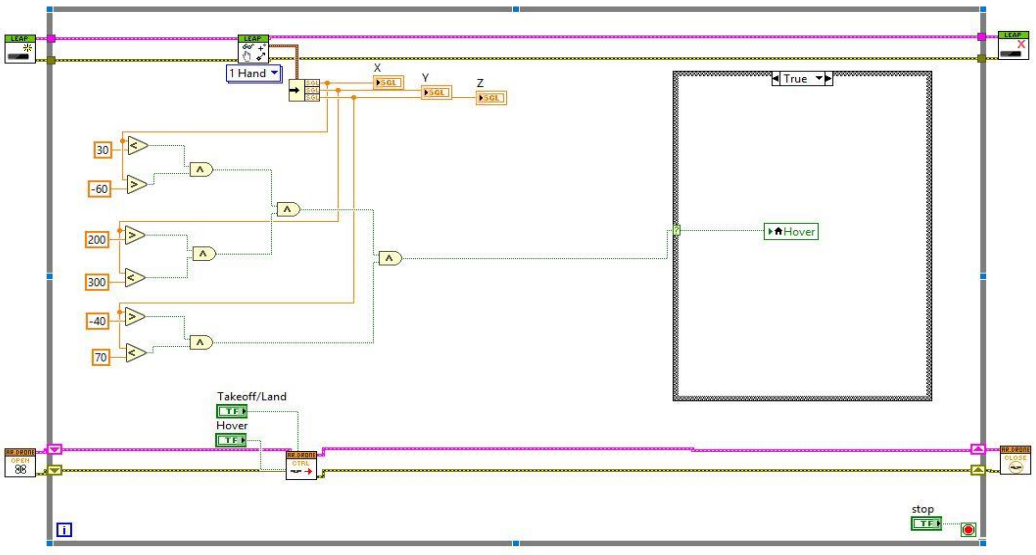
5.2.2.2 Gerakan *Hover*

Nilai kondisi dari gerakan ini tersimpan dalam variabel *hover.* Untuk baris pertama terdapat variabel *position[1]* merupakan nilai koordinat y telapak tangan dari *leap motion*. Untuk gerakan *hover* nilai koordinat y lebih dari 200 dan kurang dari 300. Untuk baris kedua terdapat variabel *position[0]* merupakan koordinat x telapak tangan dari *leap motion*. Nilai koordinat x lebih dari -60 dan kurang dari 30. Lalu baris ketiga terdapat variabel *position[2]* merupakan koordinat z telapak tangan dari *leap motion*. Nilai koordinat z lebih dari -40 dan kurang dari 70.

Kode Program 5.3 Gerakan *hover node js*

```
1  Var Hover = position[1] > 200 && position[1] < 300 &&
2  position[0] < 30 && position[0] > -60 &&
3  position[2] > -40 && position[2] < 70 &&
```

Sedangkan pada pemrograman *labview hover* juga menggunakan sudut yang sama dengan pemrograman yang menggunakan *node.js* seperti yang terlihat pada gambar 5.22 dibawah.



Gambar 5.22 Program *hover* pada *Labview*

Pada gambar diatas menggunakan sudut sama seperti pada pemrograman *node.js* menggunakan sumbu x kurang dari 30 dan x lebih dari -60 dan sumbu y

lebih dari 200 dan kurang dari 300 kemudian pada sumbu z lebih dari -40 dan kurang dari 70 maka program kan akan bernilai true dan menjalankan perintah *hover* pada *case structure* dan jika pada program diatas nilai pada tiap-tiap sumbu tidak terpenuhi maka program akan menonaktifkan perintah *hover* dan akan menjalankan perintah-perintah lain yang terdapat pada *case structure* berikutnya.

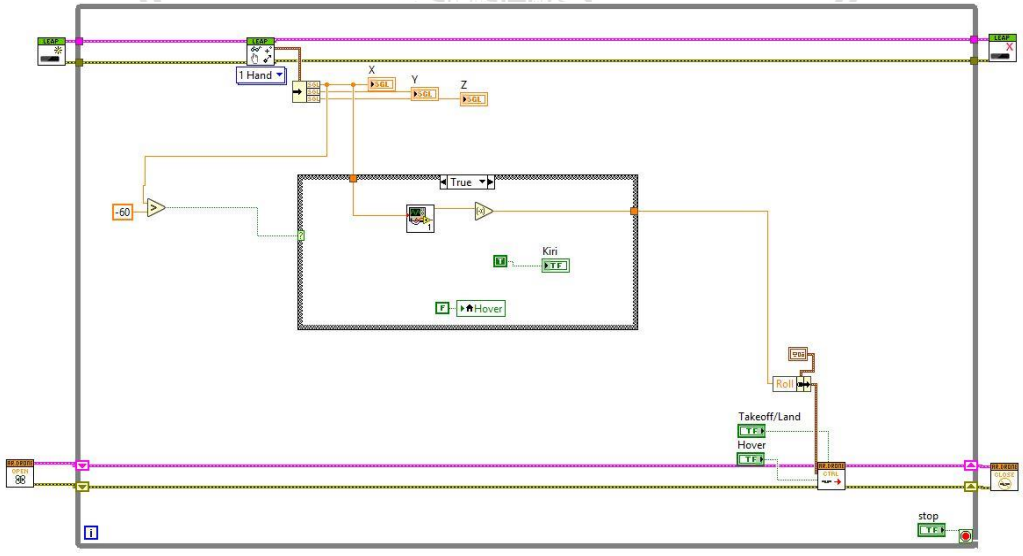
5.2.2.3 Gerakan ke Kiri

Nilai kondisi dari gerakan ini tersimpan dalam variabel *Rollleft*. Untuk baris pertama terdapat variabel *position[1]* merupakan nilai koordinat y telapak tangan dari *leap motion*. Untuk gerakan ke kiri nilai koordinat y lebih dari 200 dan kurang dari 300. Untuk baris kedua terdapat variabel *position[0]* merupakan koordinat x telapak tangan dari *leap motion*. Nilai koordinat x kurang dari -60. Lalu baris ketiga terdapat variabel *position[2]* merupakan koordinat z telapak tangan dari *leap motion*. Nilai koordinat z lebih dari -40 dan kurang dari 70.

Kode Program 5.4 Gerakan ke kiri *node js*

```
1  Var Rollleft = position[1] > 200 && position[1] < 300 &&
2                 position[0] < -60
3                 position[2] > -40 && position[2] < 70 &&
4
5
```

Sedangkan pada program *labview* untuk gerakan kiri hanya menggunakan sumbu x pada *labview* yang digunakan sebagai acuan penentuan gerakan *quadcopter* seperti pada gambar 5.23.



Gambar 5.23 Program Gerakan Ke Kiri *Labview*

Seperti yang terlihat pada gambar 5.23 dimana data posisi tangan yang terekam oleh *library leap motion* pada *labview* akan dijadikan *input*an untuk pergerakan *quadcopter*. Dimana sumbu x dari *leap motion* dihubungkan dengan persamaan kurang dari -60 yang digunakan sebagai acuan untuk menjalakan

perintah kiri. Indikator kiri akan menyalakan ketika sudut yang diinginkan memenuhusu syarat. Sumbu x dari *leap motion* akan mengirimkan data dan data akan dicek apakah lebbih dari -60 jika iya maka program bernilai true dan akan menjalankan persaam *roll*-x dimana dalam persamaan *roll*-x digunakan untuk merubah sudut menjadi kecepatan untuk *quadcopter*. Setelah hasil olahan data dari persamaan keluar data akan diinverse untuk menghasilkan nilai dari positif menjadi negative. Hal tersebut dilakukan karena pada *labview* hasil olahan data selalu menghasilkan nilai positif sedangkan yang dibutuhkan nilai negatif untuk menjalankan perintah *roll* negatif yang akan menghasilkan gerakan kiri pada *quadcopter*.

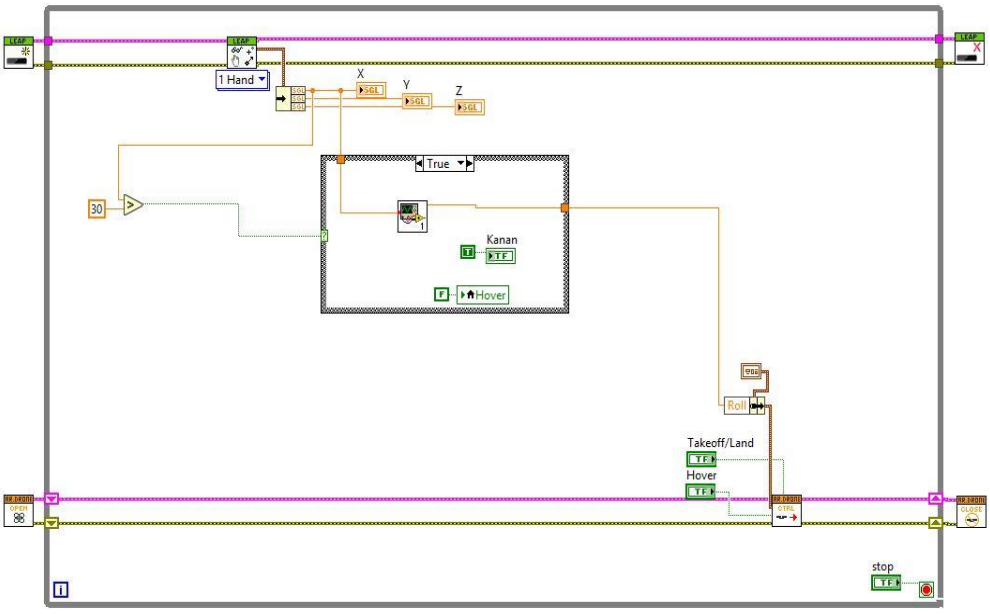
5.2.2.4 Gerakan ke Kanan

Nilai kondisi dari gerakan ini tersimpan dalam variabel *Rollright.* Untuk baris pertama terdapat variabel *position[1]* merupakan nilai koordinat y telapak tangan dari *leap motion*. Untuk gerakan ke kanan nilai koordinat y lebih dari 200 dan kurang dari 300. Untuk baris kedua terdapat variabel *position[0]* merupakan koordinat x telapak tangan dari *leap motion*. Nilai koordinat x lebih dari 30. Lalu baris ketiga terdapat variabel *position[2]* merupakan koordinat z telapak tangan dari *leap motion*. Nilai koordinat z lebih dari -40 dan kurang dari 70.

Kode Program 5.5 Gerakan ke kanan *node js*

```
1  Var Rollright = position[1] > 200 && position[1] < 300 &&
2                  position[0] > 30 &&
3                  position[2] > -40 && position[2] < 70 &&
4
5
```

Sedangkan untuk program *labview* untuk gerakna kekanan menggunakan sumbu x dan sudut lebih dari 30 seperti pada gambar 5.24.



Gambar 5.24 Program Gerakan Ke Kanan *Labview*

Seperti terlihat dari gambar 5.24 diatas terlihat data yang dihasilkan *leap motion* dihubungkan dengan persamaan lebih dari 30 jika hasil dari persamaan bernilai true maka akan menjalankan *case structure* dimana didalamnya berisi *hover* yang bernilai true, persamaan yang merubah sudut dari *leap motion* menjadi kecepatan *quadcopter* dan akan menyalakan indikator kanan pada program data hasil olahan dari persamaan langsung dikirimkan menuju kontrol cluster pada *library* control *quadcopter* untuk dijadikan *input*an gerakan kekanan pada program gerakan kekanan tidak terdapat gerbang inverse diakarenakan hasil keluaran sudah bernilai positif dan untuk menjalakan *roll* positif atau gerakan kekanan dibutuhkan nilai *output*an yang positif.

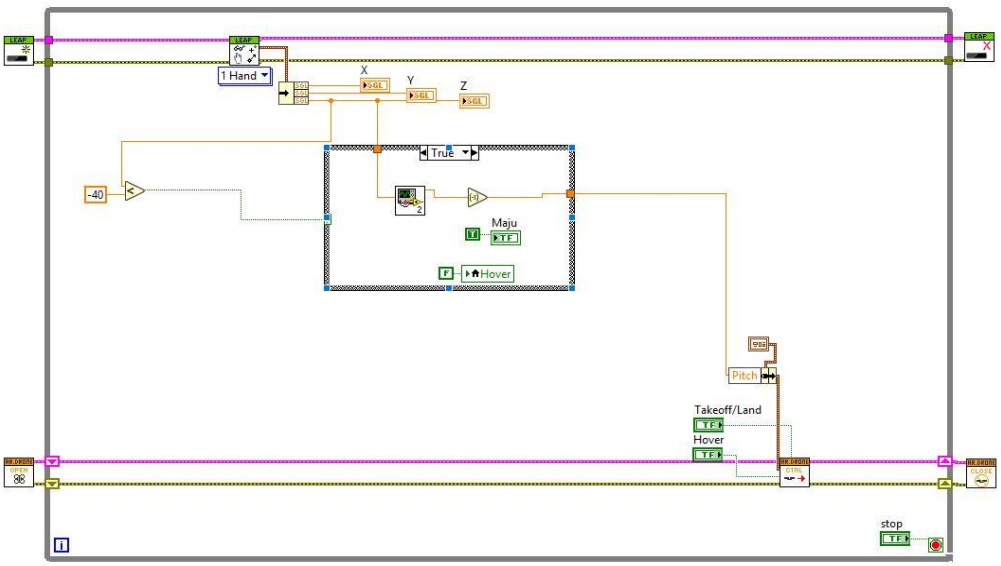
5.2.2.5 Gerakan ke Depan

Nilai kondisi dari gerakan ini tersimpan dalam variabel *Pitchfoward.* Untuk baris pertama terdapat variabel *position[1]* merupakan nilai koordinat y telapak tangan dari *leap motion*. Untuk gerakan ke depan nilai koordinat y lebih dari 200 dan kurang dari 300. Untuk baris kedua terdapat variabel *position[0]* merupakan koordinat x telapak tangan dari *leap motion*. Nilai koordinat x lebih dari -60 dan kurang dari 30. Lalu baris ketiga terdapat variabel *position[2]* merupakan koordinat z telapak tangan dari *leap motion*. Nilai koordinat z kurang dari -40.

Kode Program 5.6 Gerakan ke depan *node js*

```
1  Var Pitchfoward = position[1] > 200 && position[1] < 300 &&
2                    position[0] < 30 && position[0] > -60 &&
3                    position[2] < -40 &&
4
5
```

Sedangkan pada program *labview* digunakan 1 sumbu z saja untuk menjalaknkan gerakan ke depan seperti pada gambar 5.25.



Gambar 5.25 Program Gerakan Ke Depan *Labview*

Seperti terlihat pada gambar 5.25 gerakan kedepan pada *labview* menggunakan sumbu z dimana data dari sumbu z akan dibandingkan jika data dari sumbu z kurang dari -40 maka program bernilai true dan akan menjalankan perintah yang ada pada *case structure* menyalakan indikator Maju nilai dari sumbu z akan masuk pada persaaman *pitch*-z dan hasil dari persaam akan keluar menjadi nilai kecepatan quadopter *pitch* pada *library* control cluster quadcpoter hasil keluaran dari persamaan dimasukkan ke gerbang negate bertujuan untuk menghasilkan nilai keluaran negatif karena untuk gerakan ke depan pada *quadcopter* dibutuhkan nilai negatif.

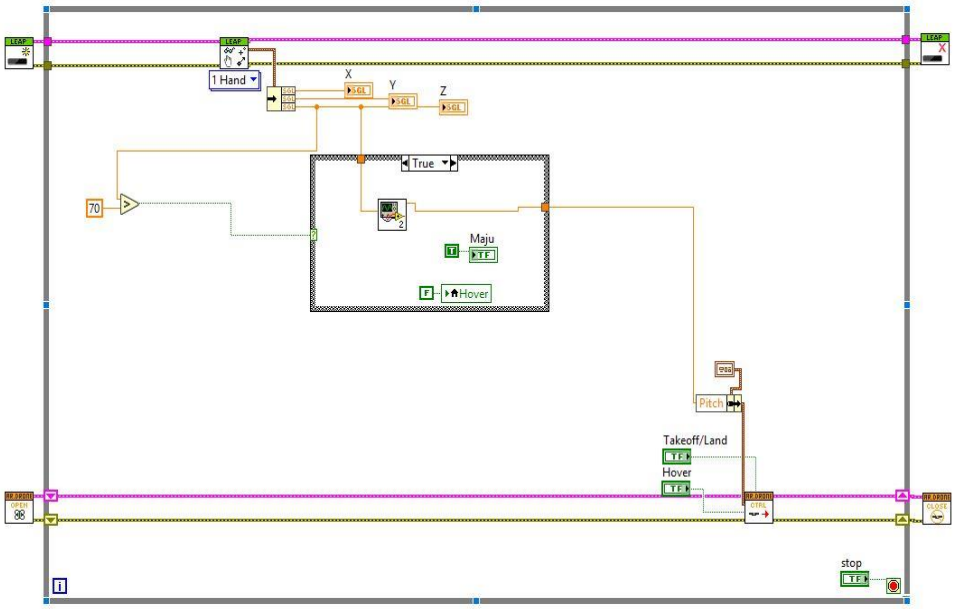
5.2.2.6 Gerakan ke Belakang

Nilai kondisi dari gerakan ini tersimpan dalam variabel *Pitchback.* Untuk baris pertama terdapat variabel *position[1]* merupakan nilai koordinat y telapak tangan dari *leap motion*. Untuk gerakan ke belakang nilai koordinat y lebih dari 200 dan kurang dari 300. Untuk baris kedua terdapat variabel *position[0]* merupakan koordinat x telapak tangan dari *leap motion*. Nilai koordinat x lebih dari -60 dan kurang dari 30. Lalu baris ketiga terdapat variabel *position[2]* merupakan koordinat z telapak tangan dari *leap motion*. Nilai koordinat z lebih dari 70.

Kode Program 5.7 Gerakan ke belakang *node js*

```
1  Var Pitchback = position[1] > 200 && position[1] < 300 &&
2                  position[0] < 30 && position[0] > -60 &&
3                  position[2] < 70 &&
4
```

Sedangkan pada program *labview* hanya menggunakan sumbu z saja sebagai syarat menjalankan gerakan ke belakang seperti pada gambar 5.26.



Gambar 5.26 Program Gerakan Ke Belakang *Labview*

Seperti yang terlihat pada gambar 5.26 *input*an nilai sudut telapak tangan yang dihasilkan *leap motion* dari sumbu z dibandingkan apakah nilai sudut lebih besar dari 70 jika iya maka hasil perbandingan akan bernilai true dan menjalakan *case structure*. Nilai dari sumbu z *leap motion* akan dimasukkan pada persamaan *pitch*-z dan hasil persamaan akan langsung dikirimkan menuju control cluster *pitch quadcopter* untuk dijadikan *input*an kecepatan *quadcopter*. Ketika nilai sumbu z lebih nadi 70 terpenuhi indikator mundur juga akan menyala bersamaan dengan *quadcopter* berjalan.

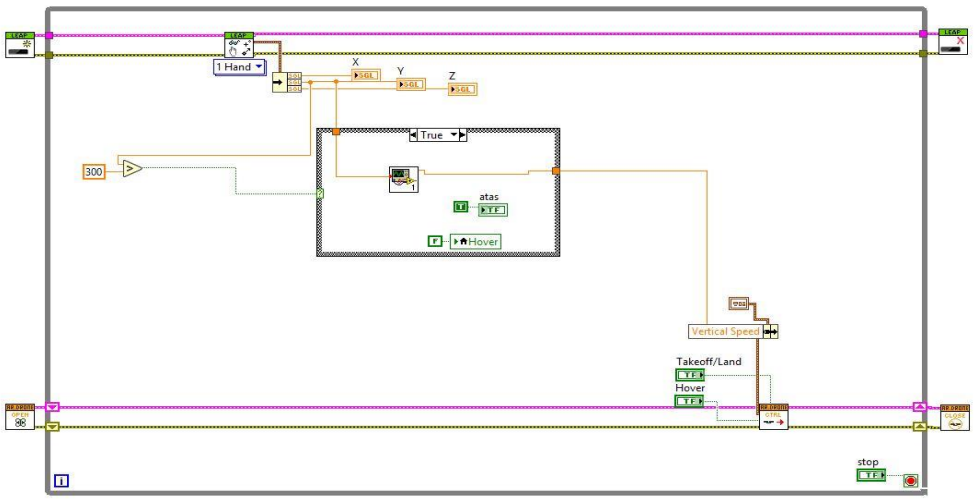
5.2.2.7 Gerakan ke Atas

Nilai kondisi dari gerakan ini tersimpan dalam variabel *Gazup.* Untuk baris pertama terdapat variabel *position[1]* merupakan nilai koordinat y telapak tangan dari *leap motion*. Untuk gerakan ke atas nilai koordinat y lebih dari 300. Untuk baris kedua terdapat variabel *position[0]* merupakan koordinat x telapak tangan dari *leap motion*. Nilai koordinat x lebih dari -60 dan kurang dari 30. Lalu baris ketiga terdapat variabel *position[2]* merupakan koordinat z telapak tangan dari *leap motion*. Nilai koordinat z lebih dari -40 dan kurang dari 70.

Kode Program 5.8 Gerakan ke atas *node js*

```
1  Var Gazup = position[1] > 300 &&
2              position[0] < 30 && position[0] > -60 &&
3              position[2] > -40 && position[2] < 70 &&
4
5
```

Sedangkan untuk gerakan keatas pada program *labview* hanya menggunakan sumbu y untuk menjalankan perintah seperti pada gambar 5.27.



Gambar 5.27 Program Gerakan Ke Atas *Labview*

Seperti yang terlihat pada gambar 5.27 dimana nilai sumbu y pada *leap motion* dibandingkan nilainya apakah lebih dari 300 jika iya maka bernilai *true* hasil perbandingan dan akan akan menjalakan *case structure*. Didalam *case structure* terdapat persamaan *gaz*-y dimana dalam persamaan yang berbentuk

sub-vi tersebut terdapat persamaan yang merubah sudut y *leap motion* menjadi *output* kecepatan pada vetical speed *quadcopter*. Hasil dari olahan data pada persamaan yang terdapat pada *case structure* akan langsung dikirm menuuju *control cluster quadcopter* untuk dijadikan *input*an kecepatan. Bersamaan itu pula indikator atas akan menyala jika nilai perbandingan y lebih dari 300 terpenuhi.

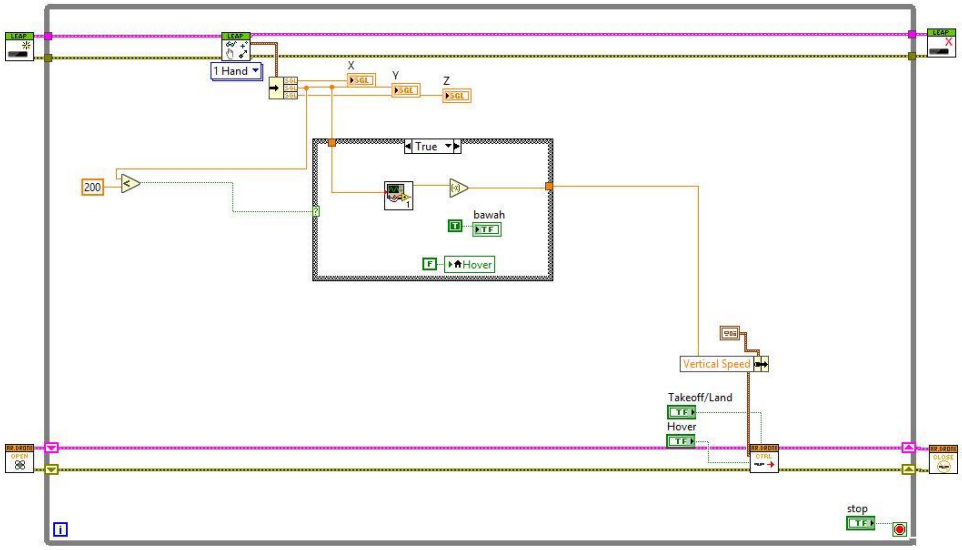
5.2.2.8 Gerakan ke Bawah

Nilai kondisi dari gerakan ini tersimpan dalam variabel *Gazdown.* Untuk baris pertama terdapat variabel *position[1]* merupakan nilai koordinat y telapak tangan dari *leap motion*. Untuk gerakan ke bawah nilai koordinat y kurang dari 200. Untuk baris kedua terdapat variabel *position[0]* merupakan koordinat x telapak tangan dari *leap motion*. Nilai koordinat x lebih dari -60 dan kurang dari 30. Lalu baris ketiga terdapat variabel *position[2]* merupakan koordinat z telapak tangan dari *leap motion*. Nilai koordinat z lebih dari -40 dan kurang dari 70.

Kode Program 5.9 Gerakan ke bawah *node js*

```
1  Var Gazdown = position[1] < 200 &&
2                position[0] < 30 && position[0] > -60 &&
3                position[2] > -40 && position[2] < 70 &&
4
5
```

Sedangkan untuk gerakan kebawah pada program *labview* hanya menggunakan sumbu y yang digunakan untuk menjakan perintah ke bawah seperti pada gambar 5.28.



Gambar 5.28 Program Gerakan Ke Bawah *labview*

Seperti yang terlihat pada gambar 5.28 hasil *output* sudut y pada *leap motion* dibandingkan apakah lebih kecil dari 200 jika iya maka akan bernilai true dan menjalakan *case structure*. Didalam *case structure* terdapat persamaan *gaz*-y

yang berbentuk sub-vi dimana berisi persamaan yang merubah sudut menjadi kecepatan. Hasil olahan dari persamaan akan dimasukkan pada gerbang negate karena dibutuhkan nilai negatif untuk gerakan kebawah pada control cluster *quadcopter*. Kemudian indikator akan menyala bersamaan dengan gerakan *quadcopter* ketika perbandingan nilai y kurang dari 200.

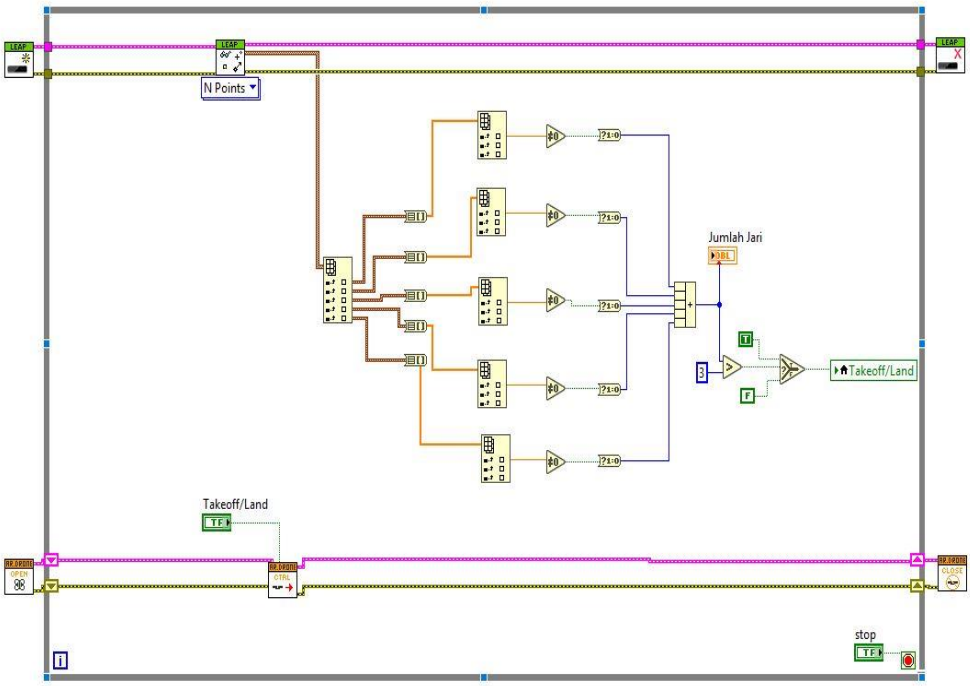
5.2.2.9 Gerakan Mendarat

Nilai kondisi dari gerakan ini tersimpan dalam salah satu *frame* dengan tipe *gesture keytap.* Pada baris kedua yaitu jika pada *frame* terdeteksi adanya tangan maka buat variabel *gesture*. Lalu baris keempat jika tipe dari *gesture* adalah *keytap* maka mengirim perintah menuju *quadcopter* untuk *hover*. Lalu pada baris 6 yaitu mengirim perintah menuju *quadcopter* untuk mendarat. Kemudian menampilkan tulisan pada *command prompt* yaitu "*Landing* the *quadcopter*".

Kode Program 5.10 Gerakan mendarat

```
1  controller.on('frame', function(frame) {
2   if(frame.gestures.length > 0){
3    var gesture = frame.gestures[0];
4    if (gesture.type == 'keyTap'){
5     client.stop();
6     client.land();
7     console.log ('Landing the quadcopter')
8    }
9   }
10 });
```

Sedangkan gerakan quadpoter pada program *labview* dibuat otomatis jika tidak terdapat titik point jari maka akan otomatis *landing* atau mendarat seperti pada gambar 5.29.



Gambar 5.29 Program Gerakan Mendarat

Seperti terlihat dari gambar 5.29 diatas gerakan mendarat *quadcopter* akan otomatis jika *leap motion* tidak mendeteksi adanya titik point jari atau titik point jari yang terdeteksi kurang dari 3 maka *quadcopter* akan secara otomatis mendarat. Itu dikarena pada *library* control *quadcopter* terdapat 2 opsi pada gerakan *takeoff* atau *landing* dimana jika *input*an pada kontol *takeoff* atau *landing* bernilai true maka *quadcopter* akan *takeoff* dan jika *input*an pada *takeoff* atau *landing* bernilai false maka kontol akan *landing*. Oleh karena itu pada program mendarat programnya sama dengan program *takeoff* hanya ketika bernilai *false* perintah mendarat akan dijalankan.

5.3 Pengiriman Data ke *Quadcopter*

5.3.1 Perancangan Pengiriman Data ke *Quadcopter*

Dalam komunikasi antara komputer dengan *quadcopter* menurut *developer guide* dari *Parrot AR Drone quadcopter* menggunakan *AT Command* yang dikirim sebagai paket *TCP* atau *UDP*. *Parrot AR Drone* mempunyai alamat IP default 192.168.1.1 dengan akses beberapa *port* untuk fungsi tertentu. *Port* 5556 berfungsi sebagai pengiriman reguler menggunakan paket *UDP*. Sedangkan *port* 5554 berfungsi sebagai membalas paket *UDP* dari *quadcopter* dan *port* 5555 berfungsi sebagai streaming data video. Panjang pengiriman *AT Command* adalah 1024 karakter. Pada Tabel 5.2 merupakan beberapa nama-nama *command* yang terdapat pada *Parrot AR Drone.*

Dalam pengendalian *quadcopter AT Command* yang digunakan *AT*PCMD* dandan *AT*REF*. *AT*PCMD* digunakan untuk sebagai menggendalikan *quadcopter* saat terbang meliputi *roll*, *pitch*, *yaw* dan *gaz*. Sedangkan *AT*REF* digunakan sebagai *takeoff*, mendarat, dan emergency reset *quadcopter*. Kedua *command* tersebut akan dijelaskan sebagai berikut.

Tabel 5.2 Daftar nama *AT Command AR Drone*

Nama *Command*	Argumen	Deskirpsi
AT*REF	*Input*	*Takeoff*/mendarat/*emergency*
AT*PCMD	*Flags, Roll, Pitch, Gaz, yaw*	Pergerakan *quadcopter*
AT*FTRIM	-	Penyetelan sudut horizontal *quadcopter*
AT*CONFIG	*Key, value*	Konfigurasi
AT*CALIB	*Device number*	Kalibrasi *magnetometer*

5.3.1.1 *AT*PCMD*

Perintah ini digunakan untuk mengendalikan gerakan *quadcopter* di udara meliputi arah gerak, kecepatan, dan rotasi. Untuk penulisan dari perintah

ini mengikuti *syntax* berikut *AT*PCMD*=[*Sequence number*],[*Flag bit-field*],[*Roll*],[*Pitch*],[*Gaz*],[*Yaw*]<LF>. Penjelasan:

a. *Sequence number* merupakan sebuah angka yang menandai urutan perintah yang sedang dijalankan. Pada saat perintah awal, *sequence number* bernilai satu, lalu untuk perintah selanjutnya nilai *sequence number* akan ditambah 1 dan seterusnya.

b. *Flag bit field integer 32 bit* merupakan *argument* untuk mengatur bagaimana *quadcopter* akan menginterpretasikan perintah progresif dari pengguna. Pada *flag field* terdapat aturan penulisan bit seperti Tabel 5.3.

Tabel 5.3 Penggunaan *bit* pada *flag field*

Bit	**31 sampai 2**	**1**	**0**
Kegunaan	Tidak digunakan	Mengaktifkan mode *combined Yaw*	Mengaktifkan perintah progresif

Pada bit 0, saat bit di atur bernilai 1 maka perintah progresif *quadcopter* yaitu *roll, pitch, yaw,* dan *gaz* akan aktif, namun pada saat bernilai 0 maka *quadcopter* akan masuk *mode hover.* Pada bit 1 saat bernilai 1 maka mode *combined yaw* akan aktif, saat bernilai 0 maka mode ini akan dimatikan. Pada bit 2 sampai 31 diatur bernilai 0 karena tidak digunakan.

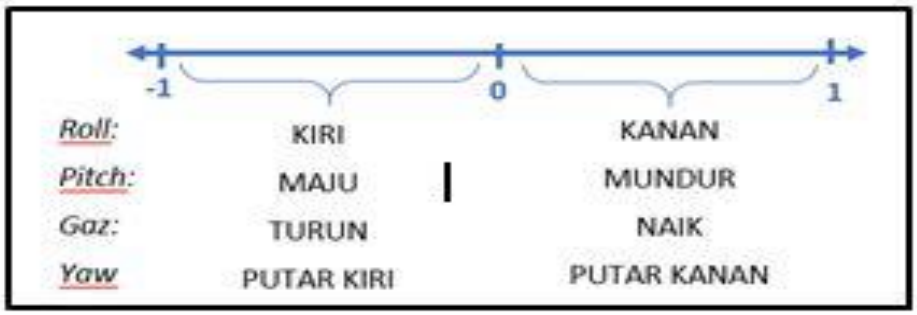
c. *Roll* digunakan untuk mengatur gerakan *quadcopter* ke kiri atau kanan beserta kecepatannya. *Argument* ini memiliki nilai dalam bentuk *floating point* antara -1 sampai 1.

d. *Pitch* digunakan untuk mengatur gerakan *quadcopter* ke depan atau belakang beserta kecepatannya. *Argument* ini memiliki nilai dalam bentuk *floating point* antara -1 sampai 1.

e. *Gaz* digunakan untuk mengatur gerakan *quadcopter* ke atas atau bawah beserta kecepatannya. *Argument* ini memiliki nilai dalam bentuk *floating* point antara -1 sampai 1.

f. *Yaw* digunakan untuk mengatur gerakan *quadcopter* berputar ke arah kiri atau kanan beserta kecepatannya. *Argument* ini memiliki nilai dalam bentuk *floating point* antara -1 sampai 1.

Dalam penulisan nilai *roll, pitch, yaw,* dan *gaz* pada *syntax* tersebut ada beberapa hal yang perlu diperhatikan. Arah pergerakan dari *quadcopter* ditentukan oleh nilai *floating point.* Pemberian nilai positif dan negatif akan mempengaruhi arah pergerakan *quadcopter* seperti pada Gambar 5.30. Sebagai contoh saat nilai dari *argument roll* diberi nilai positif maka gerakan akan ke kanan sedangkan saat diberi nilai negatif maka gerakan akan ke kiri, begitu juga dengan *argument* lainnya.



Gambar 5.30 Arah Gerakan Q*uadcopter* Berdasarkan Nilai *Floating Point*

Mengirim nilai *floating point* tersebut menggunakan *AT*PCMD*, maka nilai *floating point* perlu diubah menjadi nilai *signed hexadecimal.* Setelah itu diubah lagi menjadi nilai signed 32-bit *decimal number*. Sebagai contoh saat pengguna memberi nilai *float* 0,75 pada gerakan *roll,* maka nilai 0,75 tersebut akan diubah dahulu menjadi nilai *signed hexadecimal* yaitu 0x3f400000. Setelah itu nilai tersebut diubah lagi menjadi *decimal number* sebesar 1061158912. Sehingga pengiriman perintah *AT*PCMD* untuk gerakan *roll* ke kanan sebesar 0,75 akan menjadi seperti berikut AT* PCMD =1,1,1061158912,0,0,0<LF>

Nilai pada *floating point* tersebut merepresentasikan seberapa besar kecepatan yang akan dilakukan pergerakan *quadcopter.* Sebagai contoh *roll* sebesar 0,75 menunjukkan bahwa *quadcopter* akan bergerak ke kanan dengan kecepatan hanya 75% dari kecepatan maksimal yang bisa dijalankan oleh *quadcopter.* Pada Tabel 5.4 merupakan beberapa contoh dari nilai *roll* beserta nilai *signed hexadecimal*, signed 32-bit *decimal*, status gerakan dan kecepatan *quadcopter*.

Tabel 5.4 Contoh nilai pada gerakan *roll*

Floating point Roll	***Signed hexadecimal***	**Signed 32-bit *decimal number***	**Gerakan**	**Kecepatan**
-1	0xbf800000	3212836864	Ke kiri	Kecepatan maksimal
-0,75	0xbf400000	3208642560	Ke kiri	0,75% dari total kecepatan maksimal
-0,5	0xbf000000	3204448256	Ke kiri	0,5% dari total kecepatan maksimal
-0,25	0xbe800000	3196059648	Ke kiri	0,25% dari total kecepatan maksimal
0	0x00000000	0	Diam	Tidak ada kecepatan
0,25	0x3e800000	1048576000	Ke kanan	0,25% dari total kecepatan maksimal

0,5	0x3f000000	1056964608	Ke kanan	0,5% dari total kecepatan maksimal
0,75	0x3f400000	1061158912	Ke kanan	0,75% dari total kecepatan maksimal
1	0x3f800000	1065353216	Ke kanan	Kecepatan maksimal

5.3.1.2 *AT*REF*

Command ini digunakan untuk melakukan gerakan dasar dari *quadcopter* yaitu *Takeoff,* mendarat*,* dan *emergency reset*. Untuk penulisan dari *command* ini mengikuti *syntax* berikut AT*REF=[*Sequence number*],[*Input*]<LF>. Penjelasan:

a. *Sequence number* merupakan sebuah angka yang menandai urutan perintah yang sedang dijalankan. Pada saat perintah awal, *sequence number* bernilai satu, lalu untuk perintah selanjutnya nilai *sequence number* akan ditambah 1 dan seterusnya.

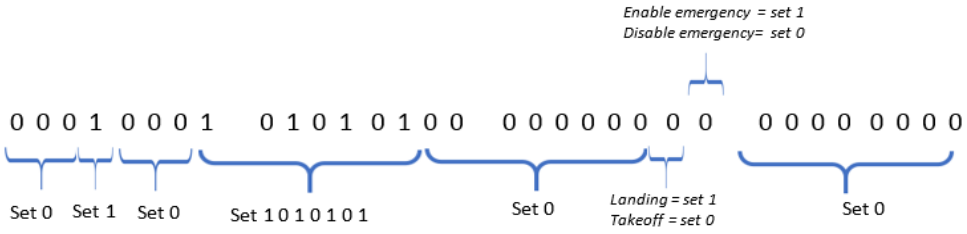
b. *Input* merupakan *argument* yang digunakan untuk mengatur gerakan dasar*quadcopter*. Format penulisannya berupa *32 bit field* seperti pada Tabel 5.5.

Tabel 5.5 Penggunaan *bit* pada *field*

Bit	**31 sampai 10**	**9**	**8**	**7 sampai 0**
Kegunaan	Tidak digunakan	*Takeoff* atau mendarat	Mode *emergency*	Tidak digunakan

Sedangkan Untuk mengirim nilai dari *input* menggunakan AT*REF, perlu diperhatikan beberapa aturan seperti yang tertera pada *developer guide* dari *Parrot AR Drone*.

Pengaturan bit dapat dilihat seperti pada Gambar 5.31. Terdapat 32 bit yang harus diatur pada *input.* Bit yang digunakan yaitu bit 8 yang berfungsi untuk mengganti mode *emergency dan* bit 9 yang berfungsi untuk *takeoff* atau mendarat selain bit tersebut tidak digunakan untuk kontrol gerakan dasar *quadcopter.* Pada bit 18, 20, 22, 24, 28 diberi nilai 1 sedangkan selain itu diberi nilai 0.



Gambar 5.31 Pengaturan Nilai Pada *Argument Input*

Setelah nilai *input* terpenuhi maka perlu diubah menjadi nilai *signed hexadecimal.* Hasil dari perubahan tersebut diubah lagi menjadi signed 32-bit *decimal number* yang nantinya nilai tersebut yang dikirim sebagai nilai *input* menggunakan AT*REF. Pada Tabel 5.6 merupakan contoh *syntax* dari AT*REF.

Tabel 5.6 Contoh *syntax AT*REF*

Binery number	*signed hexadecimal*	signed 32-bit *decimal number*	Syntax AT*REF	Gerakan
0001 0001 0101 0100 0000 0000	11540000	290717696	AT*REF=1,290717696<LF>	*Takeoff*
0001 0001 0101 0100 0000 0010	11540200	290718208	AT*REF=1,290718208<LF>	Mendarat
0001 0001 0101 0100 0000 0001	11540100	290717952	AT*REF=1,290717952<LF>	*Emergency*

5.3.2 Implementasi Pengiriman Data ke *Quadcopter*

Pada implementasi pengiriman data ke *quadcopter* ini terdapat tiga hal penting yang harus dilakukan yaitu mengubah angka *float* pada nilai kecepatan yang di masukan pengguna dari *class client.js* pada *node.js* serta control clustre pada *labview* menjadi angka *hexadecimal*, membentuk format pengiriman *AT Command atau SEND CMD* pada *labview*, serta mengkondisikan *argument* pada *AT*REF* dan *AT*PCMD*. Untuk penjelasan masing-masing fungsi tersebut akan dijelaskan sebagai berikut.

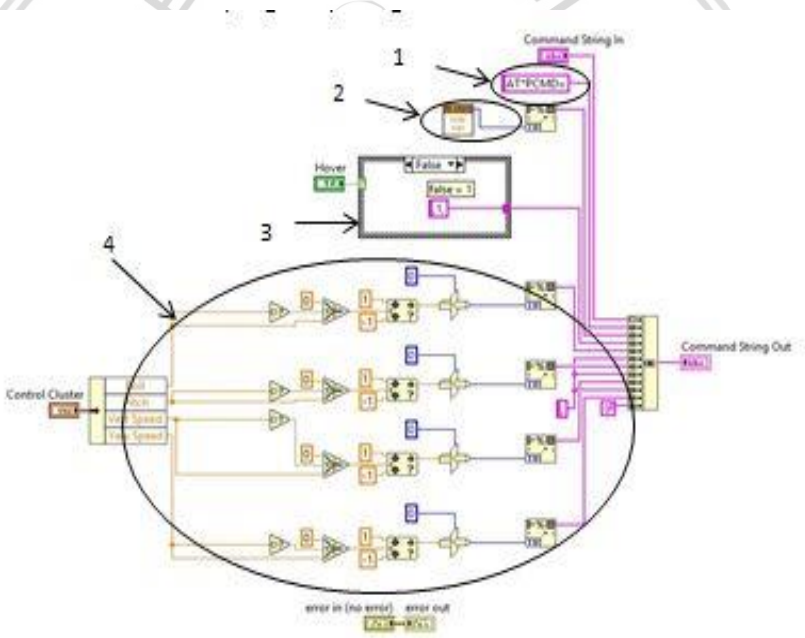
Gambar 5.32 *Library Pemrograman Labview*

1. Mengubah angka *float* mejadi *hexadecimal*. Fungsi ini digunakan untuk mengubah nilai kecepatan yang didapat dari dari *class client.js* menjadi bentuk *hexadecimal.* Fungsi ini di masukan dalam *class* at.js.

Kode Program 5.11 Nilai *float* menjadi hexadecimal

```
1  at.floatString = function(number) {
2   var buffer = new Buffer(4);
3   buffer.writeFloatBE(number, 0);
4   return -~parseInt(buffer.toString('hex'), 16) - 1;
5  };
```

Sedangkan pada pemrogram *labview input* kecepatan terdapat pada contol cluster pada *library Control Drone.vi* dimana lebih jelasnya terdapat pada *library ATPCMD.vi* seperti gambar 5.32. berikut penulis jelaskan apa proses pengolahan angka float menjadi hexadecimal untuk digunakan sebagai *input* kecapatan pada program *labview*. Contoh program pada gambar 5.33. Gambar tersebut merupakan dari peromragam *labview* yang berfungsi merubah input kecepatan dari pengguna yang berupa *floating point* menjadi *interger 32 bit* untuk dikirimkan menuju *quadcopter* sebagai input kecepatan. Setalah data input pengguna dirubah data akan langsung dikirimkan menuju *quadcopter* untuk dijadikan *input* dari gerakan. Pada gambar tersebut juga terjadi penyusunan serta pemberian nomor pada perintah atau antrian perintah yang akan dikirimkan menuju quadcpoter.



Gambar 5.33 Program *AT*PCMD*

Dari gambar 5.33 dapat dilihat format penulisan *command* yang berada pada program *AT*PCMD* dimana format penulisan sama seperti yang dijelaskan pada sub bab *AT*PCMD* dimulai dari no.1 inisialisasi penulisan *command* kemudian no.2 pemberian *sequence number*, no.3 pengaktifan perintah *hover* jika true bernilai 0 dan jika false bernilai 1. Pada bagian ini hanya berfokus menjelaskan bagian no.4 pada gambar 5.30 yaitu proses merubah *input*an *floating point* antara -1 sampai 1 dari *input*an *control cluster* dirubah menjadi integer 32 bit

setelah data dirubah menjadi integer 32 bit data akan langsung dirubah menjadi data string untuk dijadikan *input*an dalam *command* pada *AT*PCMD*.

2. Mengondisikan *argument* pada *AT Command.* Fungsi ini digunakan untuk mengatur format *argument AT*PCMD*. Baris ke 5 digunakan untuk membuat *array* yang digunakan untuk membuat *syntax* sesuai dengan pada perancangan *AT*PCMD* di bab sebelumnya. *Array* ini digunakan untuk menyimpang variabel untuk *flags, roll, pitch, gaz, yaw.* Pada baris 6-11 digunakan untuk mengondisikan nilai yang didapat dari *class client.js* menjadi nilai positif atau negatif sesuai dengan gerakan yang dilakukan. Untuk aturan kondisi terdapat pada baris 25-33. Setelah nilai dikondisikan selanjutnya nilai tersebut diubah menjadi *hexadecimal* menggunakan *class* at.js. Untuk nilai dari *flags* diatur menggunakan *bitwise* 1<<0 atau dalam angka *binary* yaitu 000000001.

Kode Program 5.12 Mengatur format *argument AT*PCMD*

```
1  AtCommandCreator.prototype.pcmd = function(options) {
2   options = options || {};
3
4  // flags, leftRight, frontBack, upDown, clockwise
5   var args = [0, 0, 0, 0, 0];
6   for (var key in options) {
7    var alias = PCMD_ALIASES[key];
8    var value = options[key];
9    if (alias.invert) {
10    value = -value;
11   }
12   args[alias.index] = at.floatString(value);
13   args[0]      = args[0] | PCMD_FLAGS.progressive;
14  }
15
16  return this.raw('PCMD', args);
17 };
18
19
20 //nilai konstan
21 var PCMD_FLAGS = exports.PCMD_FLAGS = {
22  progressive : (1 << 0),
23 };
24
25
26 var PCMD_ALIASES = exports.PCMD_ALIASES = {
27  left       : {index: 1, invert: true},
28  right      : {index: 1, invert: false},
29  front      : {index: 2, invert: true},
30  back       : {index: 2, invert: false},
31  up        : {index: 3, invert: false},
32  down       : {index: 3, invert: true},
33  clockwise    : {index: 4, invert: false},
34  counterClockwise : {index: 4, invert: true},
35 };
36
37
38
40
```

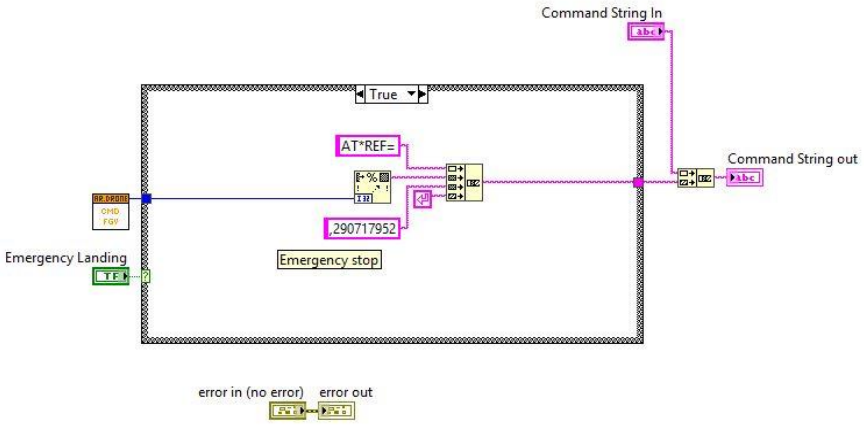
Pada program selanjutnya akan menjeleskan program AT*REF. Pada baris 1-10 merupakan fungsi untuk membentuk *argument* dari AT*REF. Saat *state* dari *class client.js* bernilai 'fly' maka akan dijalankan kode pada baris 4-5 dengan

mengambil nilai dari variabel REF_FLAG. *Takeoff* yaitu *bitwise* 1<<9 yang jika diterjemahkan dalam angka *binary* menjadi 1000000000.

Kode Program 5.13 Membentuk *argument* dari AT*REF

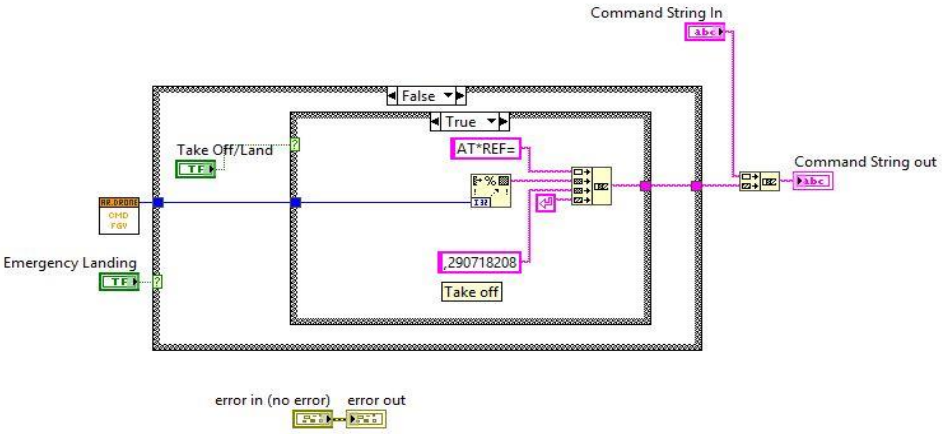
```
1  AtCommandCreator.prototype.ref = function(options) {
2   options = options || {};
3   var args = [0];
4   if (options.fly) {
5    args[0] = args[0] | REF_FLAGS.Takeoff;
6   }
7   if (options.emergency) {
8    args[0] = args[0] | REF_FLAGS.emergency;
9   }
10  return this.raw('REF', args);
11 };
12
13 var REF_FLAGS = exports.REF_FLAGS = {
14  emergency : (1 << 8),
15 Takeoff  : (1 << 9),};
16
17
```

Sedangkan program *AT*REF* pada *labview* dapat dilihat pad gambar 5.34 dibawah.



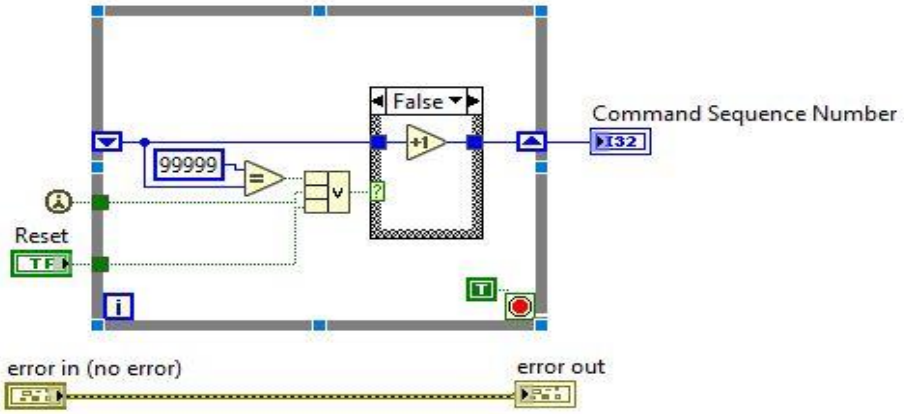
Gambar 5.34 Program *AT*REF Emergency Landing*

Pada gambar 5.34 diatas dapat dijelaskan jika didalam *library AT*REF* terdapat program dimana berisi *case structure*, *case structure* digunakan untuk membuat suatu kondisi didalam bahasa pemrograman *labview* dimana terdapat 2 kondisi pada *case structure* salah satu benilai true dan satunya lagi bernilai false. Didalam *case structure* terdapat penginisialisaian program *AT*REF* seperti ditunjuk pada no.1 selanjutnya pada no.2 yaitu sub program dari *sequence number* yang dirubah menjadi interger 32 bit untuk pemberian nomor antrian perintah, yang terakhir pada no.3 yaitu kode untuk perintah interger 32 bit yang sudah dirubah menjadi tipe data string "290717952" yang artinya emergency *landing* seperti yang berada pada tabel 5.6. Sedangkan pada *case structure* yang bernilai false contoh program dapat dilihat pada gambar 5.35 dibawah.



Gambar 5.35 Program *AT*REF Takeoff*

Pada gambar 5.35 diatas terdapat susunan yang sama dimana didalam *case structure* pertama yang bernilai false terdapat *case structure* lagi yang digunakan untuk memberikan kondisi *takeoff* dan *landing*. Dimana pada *case structure* yang bernilai true digunakan untuk mengaktifkan perintah *takeoff* dimana susuna *command* terlihat seperti gambar 5.32 diatas. Pertama inisialisasi *command AT*REF* seperti no.1 kemudian pemberian suquence number seperti no.2 untuk antrian perintah yang akan dijalankan *quadcopter* yang terakhir yaitu pemmberian *input*an interger 32 bit "290718208" yang telah dirubah menjadi tipe data string untuk dijadikan *input* perintah *takeoff* seperti pada tabel 5.6. Selanjutnya *case structure* yang bernilai false contoh program sama seperti gambar 5.33 struktur susunan program sama hanya pada no.3 berisi "290717696" yang artinya mengaktifkan perintah *landing*. Sedangkan untuk program *sequence number* contoh gambar terdapat pada gambar 5.36 dibawah.



Gambar 5.36 Program *Sequence Number*

Pada program diatas digunakan untuk penomoran perintah pada tiap-tiap *command* yang akan dijalankan. Penomoran dijalankan dengan cara terus

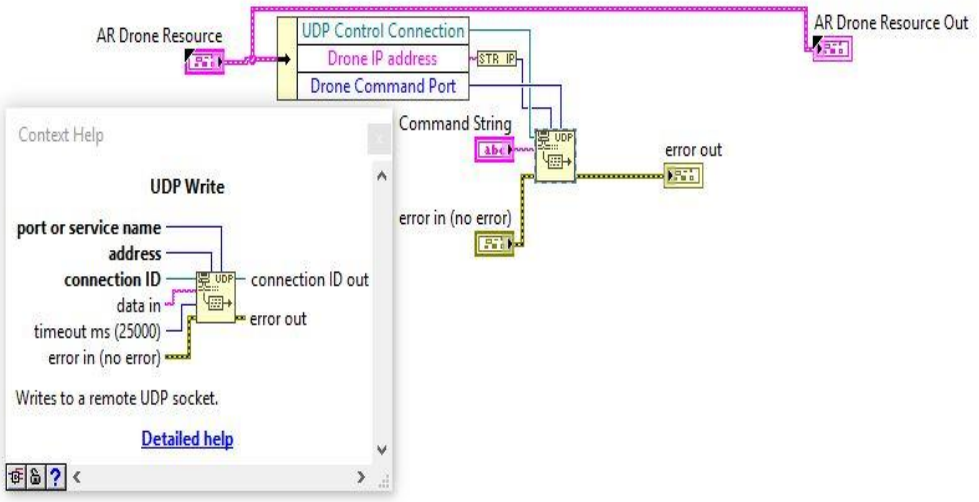
menambah 1 setiap dijalankan dan akan direset ketika data antrian mencapai 99999.

3. Membentuk format pengiriman *AT Command*. Fungsi ini digunakan untuk membentuk format *AT Command* yang akan dikirim. *this.type* berarti fungsi ini memberikan AT apa yang dikirim apakah REF atau PCMD. Untuk *args.joint* berisi nilai *sequence number* ditambah dengan *argument* dimasing-masing *AT*REF* atau *AT*PCMD*. Fungsi ini terdapat pada *class* At*Command*.js

Kode Program 5.14 Format pengiriman *AT Command*

```
1  AtCommand.prototype.serialize = function(sequenceNumber) {
2   var args = [sequenceNumber].concat(this.args);
3   return 'AT*' + this.type + '=' + args.join(',') + '\r';
4  };
```

Sedangkan pada pemrogram *labview* untuk pengiriman AT *Command* berada pada *library* SEND CMD seperti pada gambar 5.37 dibawah.



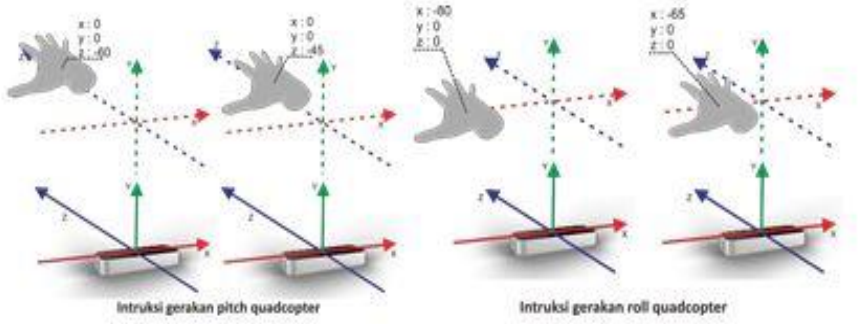
Gambar 5.37 Program *Send Cmd*

Pada gambar 5.37 dapat dijelaskan pada program SEND CMD terdapat *UDP* control *Connection*, *Drone Ip address dan Drone Command port* serta *command* string untuk meng*input*kan *command-command* yang telah berada dalam antrian *sequence number*. Pada gambar 5.37 diatas juga diatas telah diset *default* ketika mencapai 25000 ms atau 25 second tidak terdapat perintah maka sistem akan otomatis *timeout*.

5.4 Fungsi Kecepatan

5.4.1 Perancangan Fungsi Kecepatan

Pada sub bab ini dilakukan perancangan fungsi kecepatan dari *quadcopter*. Perancangan ini bertujuan untuk menghasilkan kecepatan berbeda setiap gerakan tangan yang dilakukan oleh pengguna. Nilai yang digunakan untuk parameter kecepatan *quadcopter* menggunakan nilai koordinat yang dihasilkan dalam gerakan tangan pengguna. Pada Gambar 5.38 terlihat dua variasi gerakan yaitu gerakan tangan ke depan untuk mengendalikan *quadcopter* bergerak ke depan dan gerakan tangan ke kiri untuk mengendalikan *quadcopter* bergerak ke kiri. Pada saat tangan pengguna bergerak ke depan sedikit maka nilai koordinat z menjadi -45 dan ketika tangan pengguna bergerak kedepan dengan jauh maka nilai koordinat z menjadi -60. Dengan melihat perbedaan nilai tersebut maka dibuatlah sebuah fungsi agar pengguna dapat mengatur kecepatan *quadcopter* yang diinginkan. Jika pengguna ingin mengendalikan *quadcopter* bergerak ke depan dengan cepat maka pengguna harus menggerakkan tangannya ke depandengan jauh sedangkan pengguna ingin mengendalikan *quadcopter* bergerak ke depan dengan pelan maka pengguna harus menggerakkan tangannya ke depan dengan sedikit begitu juga dengan gerakan yang lain. Untuk menghasilkan fungsi tersebut maka diperlukan perhitungan berdasarkan nilai koordinat tiap gerakan.



Gambar 5.38 Nilai Koordinat Pada Gerakan Ke Depan Dan Ke Kiri

Nilai koordinat yang digunakan mengikuti dari masing-masing gerakan. Sebagai contoh untuk gerakan *roll* maka nilai yang digunakan yaitu nilai koordinat x *leap motion*. Pada gerakan *pitch* maka nilai yang digunakan yaitu nilai koordinat z *leap motion*. G erakan *gaz* maka nilai yang digunakan yaitu nilai koordinat y *leap motion*. Hasil dari perhitungan ini dikirim menjadi nilai kecepatan pada *AT*PCMD*. Dalam ketentuan dari *AT*PCMD* nilai yang dapat diterima adalah antara 0 sampai 1 pada pemrograman *node.js* dan nilai -1 sampai 1 pada pemrograman *labview* untuk setiap gerakan sehingga hasil dari perhitungan kecepatan harus menghasilkan nilai tersebut yang bisa ditrima oleh *AT*PCMD*. Nilai kecepatan dalam fungsi ini berbentuk *float*. Maka dibutuhkan sebuah persamaan untuk membuat fungsi kecepatan ini. Persamaan yang digunakan dalam kecepatan didapatkan dalam pengamatan dan percobaan terhadap nilai koordinat setiap gerakan pada *leap motion*. Hasil percobaan dapat

dilihat pada sub bab 6.4. Untuk perhitungan nilai kecepatan tiap gerakan akan dijelaskan sebagai berikut:

a. Perhitungan nilai kecepatan gerakan *roll*. Dalam gerakan *roll* mencakup dua kondisi yaitu gerakan ke kanan dan gerakan ke kiri. Parameter yang digunakan yaitu nilai koordinat x pada *leap motion*. Perhitungan dari nilai *input* kecepatan *roll (Vroll)* dapat dilihat pada Persamaan 5.2. Perhitungan ini didapat dari percobaan berulang kali peneliti untuk mengubah nilai koordinat yang dihasilkan *leap motion* untuk menjadi nilai *input* untuk kecepatan *quadcopter*. Nilai koordinat x pada *leap motion (PositionX)* dibagi dengan nilai maksimal dari nilai koordinat x pada *leap motion (MaxPositionX).* Dalam hal ini nilai maksimalnya adalah 500 untuk gerakan *roll*. Nilai dari *Vroll* diabsolutkan untuk menghindari adanya nilai negatif dikarenakan pada *output* dari *AT*PCMD* nilainya adalah dari 0 sampai 1 pada pemrograman *node.js* sedangkan pada pemrograman *labview output* yang dihasilkan antara -1 sampai 1.

$$Vroll = \frac{\text{positionx}}{\text{maxpositiox}} \quad (5.1)$$

Contohnya:

$$Vroll = \frac{326}{500}$$
$$Vroll = 0{,}652$$

Dari contoh tersebut dapat dilihat hasil kecepatan yang didapat pada posisi tangan 326 mm pada sumbu x *leap motion* menghasilkan kecepatan 0,652 pada gerakan *roll* dan gerakan *quadcopter* akan ke kanan sebaliknya jika *output* dari *vroll* bernilai negatif maka *quadcopter* akan bergerak ke kiri.

b. Perhitungan nilai *input* kecepatan gerakan *pitch*. Dalam gerakan *pitch* mencakup dua kondisi yaitu gerakan ke depan dan gerakan ke belakang. Parameter yang digunakan yaitu nilai koordinat z pada *leap motion*. Perhitungan dari nilai *input* kecepatan *pitch (Vpitch)* dapat dilihat pada Persamaan 5.1. Perhitungan ini didapat dari percobaan berulang kali peneliti untuk mengubah nilai koordinat yang dihasilkan *leap motion* untuk menjadi nilai *input* untuk kecepatan *quadcopter*. Nilai koordinat z pada *leap motion (PositionZ)* dibagi dengan nilai maksimal dari nilai koordinat z pada *leap motion (MaxPositionZ).* Dalam hal ini nilai maksimalnya adalah 400 untuk gerakan *Pitch*. Nilai dari *Vpitch* diabsolutkan untuk menghindari adanya nilai negatif dikarenakan pada *output* dari *AT*PCMD* nilainya adalah dari 0 sampai 1 pada pemrograman *node.js* sedangkan pada pemrograman *labview* adalah -1 sampai 1.

$$\text{V}pitch = \frac{\text{positionz}}{\text{maxpositionz}} \quad (5.2)$$

Contohnya:

$$Vpitch = \frac{-225}{400}$$

$$Vpitch = -0{,}5625$$

Dari contoh tersebut dapat dilihat hasil kecepatan yang didapat pada posisi tangan -225 mm pada sumbu z *leap motion* menghasilkan kecepatan -0,5625 pada gerakan *pitch* dan gerakan *quadcopter* akan ke depan begitu sebaliknya jika *output* dari *vpitch* bernilai positif maka *quadcopter* akan bergerak ke belakang.

c. Perhitungan kecepatan gerakan *gaz*. Dalam gerakan *gaz* mencakup dua kondisi yaitu gerakan ke atas dan gerakan ke bawah. Parameter yang digunakan yaitu nilai koordinat y pada *leap motion*. Perhitungan dari nilai *input* kecepatan *gaz (Vgaz)* dapat dilihat pada Persamaan 5.4. Perhitungan ini didapat dari percobaan berulang kali peneliti untuk mengubah nilai koordinat yang dihasilkan *leap motion* untuk menjadi nilai *input* untuk kecepatan *quadcopter*. Nilai koordinat y pada *leap motion (PositionY)* dibagi dengan nilai maksimal dari nilai koordinat y pada *leap motion (MaxPositionY).* Dalam hal ini nilai maksimalnya adalah 700 untuk gerakan *gaz*. Dikarenakan pada nilai koordinat *PositionY* untuk gerakan *gaz* keseluruhan bernilai positif dan berdampak terhadap gerakan *gaz* ke bawah yaitu nilai V*gaz* semakin kebawah semakin berkurang maka hasil dari pembagian *PositionY* dengan *MaxPositionY* dikurangi dengan nilai antara 0 sampai 1. Dalam sistem ini menggunakan nilai 0,5 dikarenakan telah sesuai dengan kebutuahan. Nilai dari *Vgaz* diabsolutkan untuk menghindari adanya nilai negatif dikarenakan pada *output* dari *AT*PCMD* nilainya adalah dari 0 sampai 1 pada pemrograman *node.js* sedangkan pada pemrograman *labview output* yang dihasilkan antara -1 sampai 1.

$$\mathrm{V}gaz = \frac{\mathrm{positionY}}{\mathrm{maxpositionY}} \quad (5.3)$$

$$Vgaz = \frac{420}{700}$$

$$Vgaz = 0{,}6$$

Dari contoh tersebut dapat dilihat hasil kecepatan yang didapat pada posisi tangan 420 mm pada sumbu y *leap motion* menghasilkan kecepatan -0,6 pada gerakan *gaz* dan gerakan *quadcopter* akan ke atas begitu sebaliknya jika *output* dari *vpitch* bernilai negatif maka *quadcopter* akan bergerak ke bawah.

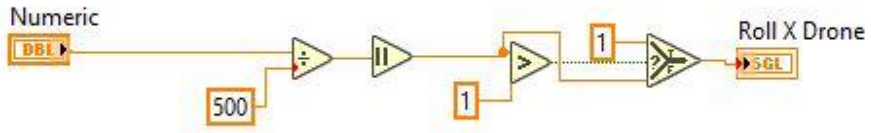
5.4.2 Implementasi Fungsi Kecepatan

1. Perhitungan nilai kecepatan gerakan *roll*. Dalam fungsi ini digunakan satu parameter yaitu *positionx.* Variabel *position*x dibagi dengan 500 yaitu nilai maksimal dari nilai koordinat x pada *leap motion*. Lalu nilai *Vroll* diabsolutkan sehinga menghasilkan nilai *float* positif dan dilakukan *return*. Jika nilai dari *Vroll* lebih dari satu maka nilai *Vroll* diubah menjadi 1 sedangkan jika nilai dari *Vroll* kurang dari satu maka nilai *Vroll* tetap sama sesuai perhitungan.

Kode Program 5.15 Fungsi nilai *input* kecepatan gerakan *roll*

```
1  function kecepatan_Roll(positionx){
2  var Vroll= Math.abs(positionz /500);
3  return (Vroll>1)? 1:Vroll;
4  }
```

Sedangkan untuk contoh kode program dari fungsi kecepatan gerakan *roll* pada *labview* dapat dilihat dari gambar 5.39 untuk fungsi penentuan kecepatan sendiri sama antara pemrograman *labview* dengan *node.js* sama.



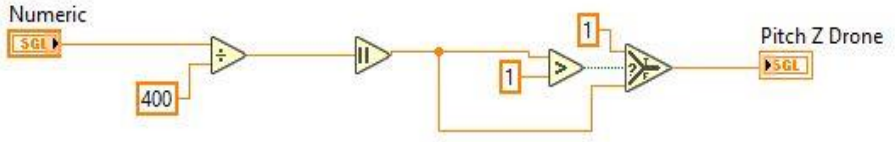
Gambar 5.39 Fungsi Kecepatan Gerakan *Roll*

2. Perhitungan nilai *input* kecepatan gerakan *pitch*. Dalam fungsi ini digunakan satu parameter yaitu *positionz.* Variabel *positionz* dibagi dengan 400 yaitu nilai maksimal dari nilai koordinat z pada *leap motion*. Lalu nilai *Vpitch* diabsolutkan sehinga menghasilkan nilai *float* positif dan dilakukan *return*. Jika nilai dari *Vpitch* lebih dari satu maka nilai *Vpitch* diubah menjadi 1 sedangkan jika nilai dari *Vpitch* kurang dari satu maka nilai *Vpitch* tetap sama sesuai perhitungan.

Kode Program 5.16 Fungsi nilai *input* kecepatan gerakan *pitch*

```
1  function kecepatan_Pitch(positionz){
2        var Vpitch= Math.abs(positionz /400);
3        return (Vpitch>1)? 1:Vpitch;
4  }
```

Sedangkan untuk contoh kode program dari fungsi kecepatan gerakan picth *labview* dapat dilihat dari gambar 5.40 untuk fungsi penentuan kecepatan sendiri sama antara pemrograman *labview* dengan *node.js* sama.



Gambar 5.40 Fungsi Kecepatan Gerakan *Pitch*

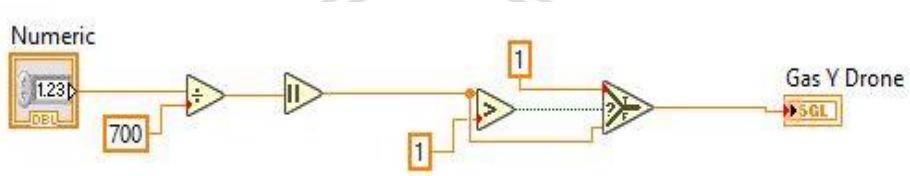
3. Perhitingan kecepatan gerakan *gaz*. Dalam fungsi ini digunakan satu parameter yaitu *positiony*. Variabel *positiony* dibagi dengan 700 yaitu nilai maksimal dari nilai koordinat y pada *leap motion*. Lalu hasil dari pembagian tersebut digurangi dengan 0,5. Setelah itu nilai *Vgaz* diabsolutkan sehinga menghasilkan nilai *float* positif dan dilakukan *return*.

Jika nilai dari *Vgaz* lebih dari satu maka nilai *Vgaz* diubah menjadi 1 sedangkan jika nilai dari *Vgaz* kurang dari satu maka nilai *Vgaz* tetap sama sesuai perhitungan.

Kode Program 5.17 Fungsi nilai *input* kecepatan gerakan *gaz*

```
1   function kecepatan_Gaz(positiony){
2         var Vgaz= Math.abs(positiony /700 -0.5);
3         return (Vgaz>1)? 1:Vgaz;
4       }
```

Sedangkan untuk contoh kode program dari fungsi kecepatan gerakan *gaz* atau vertical speed pada *labview* dapat dilihat dari gambar 5.41 untuk fungsi penentuan kecepatan sendiri sama antara pemrograman *labview* dengan *node.js* sama.



Gambar 5.41 Fungsi Kecepatan *Gaz* Atau *Vertical Speed*

BAB 6 PENGUJIAN DAN ANALISIS

Bagian ini memuat kesimpulan dan saran terhadap skripsi. Kesimpulan dan saran disajikan secara terpisah, dengan penjelasan sebagai berikut:

6.1 Pengujian Nilai Koordinat Gerakan Kendali *Quadcopter*

6.1.1 Tujuan Pengujian

Pengujian ini bertujuan untuk mengetahui batasan-batasan koordinat yang akan dijadikan pedoman penetuan batasan koordinat untuk perintah gerakan pada *quadcopter* sesuai dengan yang telah dirancang atau tidak.

6.1.2 Pelaksanaan Pengujian

Pengujian nilai koordinat gerakan kendali *quadcopter* dilakukan dengan cara menggerakkan tangan diatas leap dan melihat hasil koordinat yang telah ditentukan sesuai atau tidak. Gerakan *pitch* pertama memposisikan tangan tepat diatas *leap motion* kemudian menggerakkan tangan ke arah belakang dan ke arah depan *leap motion*. Pada gerakan *roll* dilakukan dengan cara memposisikan tangan tepat diatas *leap motion* kemudian. Pada gearakan *gaz* dilakukan dengan cara memposisikan tangan diatas *leap motion* kemudian bergerak ke bawah menuju *leap motion* dan ke atas menjahui *leap motion*. Untuk gerakan *takeoff*, *hover*, *landing* dilakukan dengan cara memposisikan tangan berada tepat pada diatas *leap motion* usahakan jari melebar karena syarat untuk gerakan *takeoff* butuh deteksi jari minimal 4 jari terdeteksi, telapak tangan membuka untuk gerakan *hover* dan pindahkan posisi tangan dari *leap motion* agar tidak terdeteksi dan quadcpoter akan otomatis mendarat. Untuk setiap gerakan dilakukan 5 kali percobaan. Dari hasil percobaan tersebut maka dijadikan pedoman untuk menentukan koordinat pada setiap gerakan tangan yang digunakan untuk mengendalikan *quadcopter* pada sub bab 5.2.

6.1.3 Langkah-Langkah Pengujian

Berikut merupakan langkah-langkah yang akan dilakukan dalam pengujian:

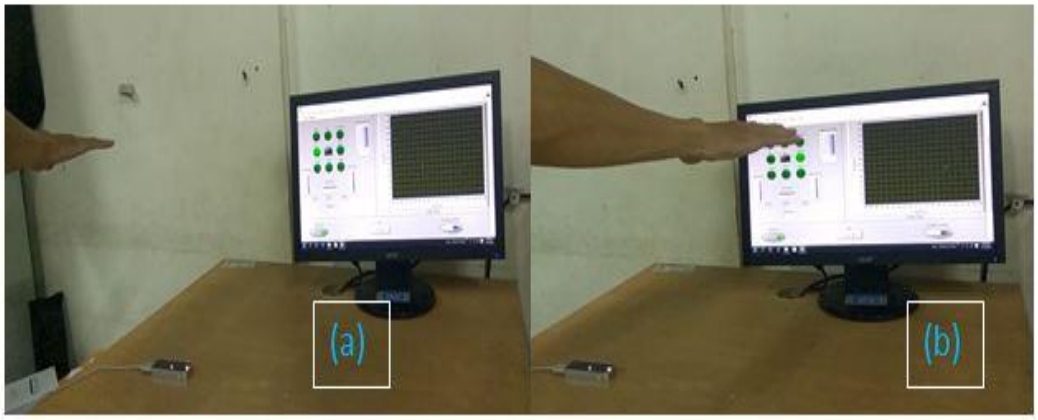
1. *Leap motion* dihubungkan dengan komputer menggunakan kabel *USB.*
2. Membuka aplikasi *leap motion* control panel. Lalu muncul *icon leap motion* pada di pojok kanan bawah destop, jika *icon leap motion* berwarna hijau maka *leap motion* siap digunakan. Jika *icon leap motion* berwarna hitam maka klik kanan pada *icon* tersebut lalu pilih *resume tracking*. Jika *icon* berwarna *orange* maka *leap motion* perlu dibersihkan permukaannya terlebih dahulu sebelum digunakan.
3. Membuka *command prompt node js* pada komputer lalu menjalankan program "*node* sistemkontrol*quadcopter*.js" untuk program *node.js* atau membuka *labview* dan jalankan project uji dan double click pada program "*coba uji.vi*".

4. Pengguna memposisikan tangan di atas *leap motion.*
5. Menggerakkan tangan sesuai dengan intruksi yang telah ditentukan yaitu gerakan *pitch*, *roll*, *gaz*, *takeoff*, *hover*, mendarat.
6. Melihat nilai koordinat yang dihasilkan pada setiap gerakan pada *command prompt* atau *indikator* pada *labview* yang sudah ada dan nilai koordinat tersebut disalin kedalam *microsoft excel* untuk diolah untuk dijadikan pedoman pada perancangan gerakan.
7. Mengulangi langkah ke-5 sebanyak lima kali untuk oleh pengguna.

6.1.4 Hasil Pengujian

6.1.4.1 Gerakan *Roll*

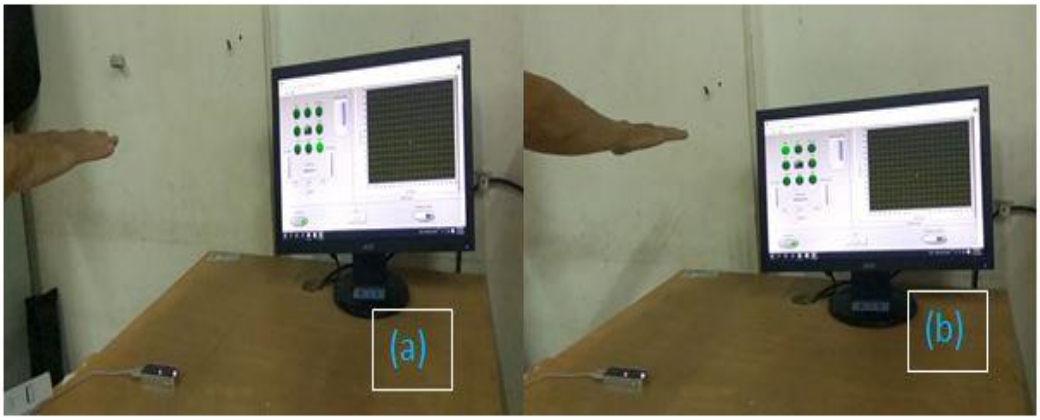
Pada Gambar 6.1 dapat dilihat proses melakukan pengujian gerakan *roll*. Pada bagian (a) merupakan gerakan *roll* ke kiri yaitu menggerakkan tangan dengan posisi awal tangan tepat berada diatas *leap motion* lalu tangan bergerak ke kiri sampai *leap motion* tidak mendeteksi tangan pengguna. Bagian (b) merupakan gerakan *roll* ke kanan yaitu menggerakkan tangan dengan posisi awal tangan tepat berada diatas *leap motion* lalu tangan bergerak ke kanan sampai *leap motion* tidak mendeteksi tangan pengguna.



Gambar 6.1 Pengujian Gerakan *Roll*

6.1.4.2 Gerakan *Pitch*

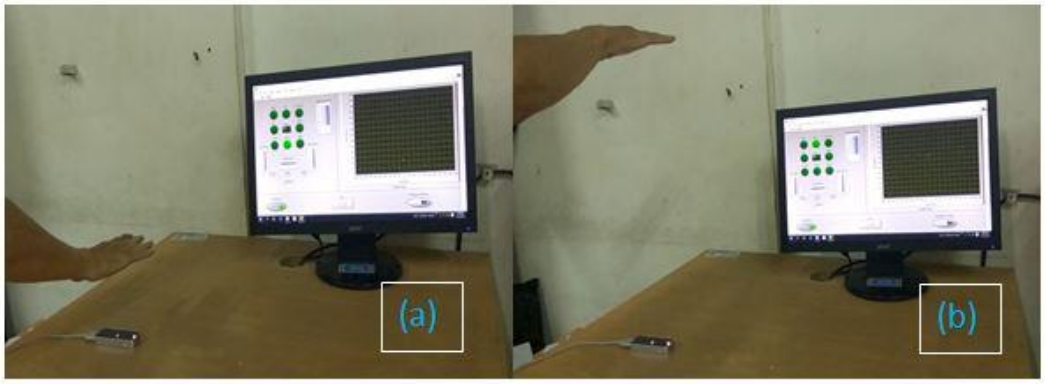
Pada Gambar 6.2 dapat dilihat proses melakukan pengujian gerakan *pitch*. Pada bagian (a) merupakan gerakan *pitch* ke belakang yaitu menggerakkan tangan dengan posisi awal tangan tepat berada diatas *leap motion* lalu tangan bergerak ke belakang sampai *leap motion* tidak mendeteksi tangan pengguna. Bagian (b) merupakan gerakan *pitch* ke depan yaitu menggerakkan tangan dengan posisi awal tangan tepat berada diatas *leap motion* lalu tangan bergerak ke depan sampai *leap motion* tidak mendeteksi tangan pengguna.



Gambar 6.2 Pengujian Gerakan *Pitch*

6.1.4.3 Gerakan *Gaz*

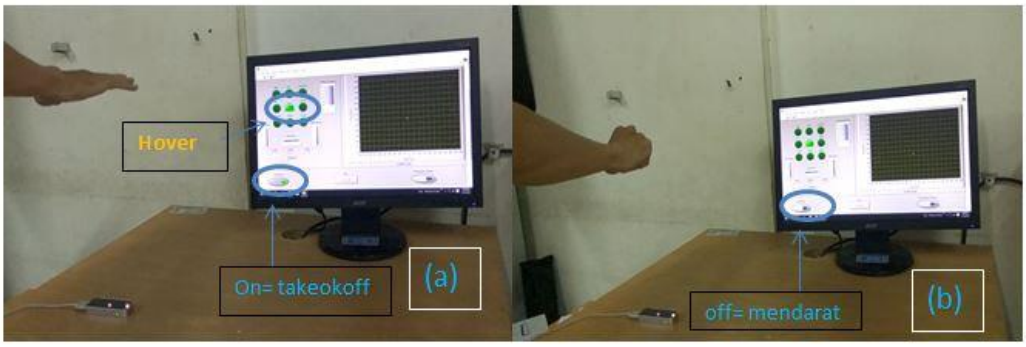
Pada Gambar 6.3 dapat dilihat proses melakukan pengujian gerakan *gaz*. Pada bagian (a) merupakan gerakan *gaz* ke bawah yaitu menggerakkan tangan dengan posisi awal tangan tepat berada diatas *leap motion* lalu tangan bergerak ke bawah sampai *leap motion* tidak mendeteksi tangan pengguna. Bagian (b) merupakan gerakan *gaz* ke atas yaitu menggerakkan tangan dengan posisi awal tangan tepat berada diatas *leap motion* lalu tangan bergerak ke atas sampai *leap motion* tidak mendeteksi tangan pengguna.



Gambar 6.3 Pengujian Gerakan *Gaz*

6.1.4.4 Gerakan *Takeoff, Hover dan Landing*

Pada Gambar 6.4 dapat dilihat proses melakukan pengujian gerakan *takeoff, hover* dan mendarat. Pada bagian (a) merupakan gerakan *takeoff* yaitu telapak tangan dengan posisi awal tangan tepat berada diatas *leap motion* lalu telapak tangan merenggangkan jari-jari pada posisi *hover*. Bagian (b) posisi tangan tetap pada posisi *hover* tetapi posisi jari digenggam atau dengan posisi jari yang rapat supaya hal tersebut akan mengakibatkan perintah mendarat dijalankan, dikarenakan pada sistem yang telah dibuat di *labview* gerakan *takeoff* akan berjalan ketika sistem mendeteksi jumlah jari lebih dari 3 jika sistem tidak mendeteksi jri maka otomatis mendarat.



Gambar 6.4 Pengujian *Takeoff, Hover*, Mendarat

6.1.4.5 Analisis Hasil Pengujian

Dari pengujian dihasilkan koordinat-koordinat dari data navigasi *quadcopter*. Data koordinat tersebut dikolompokkan berdasarkan gerakan *quadcopter*.

Tabel 6.1 Hasil percobaan penentuan koordinat gerakan *pitch* pada *node js*

Gerakan (*pitch*)	Nilai		
	Koordinat x (mm)	Koordinat y (mm)	Koordinat z (mm)
Ke belakang	-16,671	249,6388	75,71804
Ke belakang	-11,7873	253,8842	166,6108
Ke belakang	-10,9168	257,133	242,595
Ke belakang	-6,39582	260,8422	303,4686
Ke belakang	-0,30152	270,322	393,812
Ke depan	-16,7511	251,413	-48,5957
Ke depan	-23,3282	264,908	-302,02
Ke depan	-23,705	261,714	-332,279
Ke depan	-23,3597	256,1836	-358,004
Ke depan	-28,1759	256,627	-385,824

Pada Tabel 6.1 merupakan tabel dari nilai koordinat yang dihasilkan dari gerakan *pitch*. Nilai koordinat x, y mengalami perubahan yang tidak tetap. Sedankan nilai koordinat z mengalami perubahan yang tetap yaitu semakin bertambah dan semakin berkurang dikarenakan untuk gerakan *pitch leap motion* menggunakan koordinat z sebagai tumpuan gerakan hal tersebut sama halnya dengan koordinat yang ada pada program *labview* Seperti pada Tabel 6.2. Sehingga pada gerakan *pitch* diambil nilai koordinat z untuk pedoman menentukan batasan gerakan *pitch* dengan *hover* untuk koordinat z pada *leap motion*.

Pada tabel 6.2 dibawah juga mengalami hal sama dimana nilai pada sumbu x,y mengalam perubah yang tidak tetap, sedangkan pada sumbu z mengalami perubahan yang tetap dimana nilai negatif digunakan untuk gerakan kedepan *quadcopter* dan nilai positif digunakan untuk gerakan kebelakang.

Tabel 6.2 Hasil Percobaan Penentuan Koordinat Gerakan *Pitch* Pada *Labview*

Gerakan (*pitch*)	Nilai		
	Koordinat x (mm)	Koordinat y (mm)	Koordinat z (mm)
Ke belakang	28,0549	206,4542	79,89106
Ke belakang	18,3059	224,6257	155,81728
Ke belakang	20,5144	230,3578	240,60899
Ke belakang	2,0142	237,3735	334,19291
Ke belakang	0,2878	240,5835	367,89381
Ke depan	17,2726	226,3484	-56,96166
Ke depan	21,3788	182,6058	-134,28067
Ke depan	18,7486	226,7718	-194,25013
Ke depan	15,6029	229,0578	-230,01307
Ke depan	13,0251	230,3578	-270,46708

Pada Tabel 6.3 merupakan nilai koordinat yang dihasilkan pada gerakan *roll*. Nilai koordinat y, z mengalami perubahan yang tidak tetap. Sedangkan nilai koordinat x mengalami perubahan yang tetap dikarenakan untuk gerakan *roll leap motion* menggunakan sumbu x sebagai tumpuan gerakan tersebut. Sama halnya dengan koordinat yang ada pada program *labview* seperti pada Tabel 6.4 sehingga pada gerakan *roll* diambil nilai koordinat x untuk pedoman menentukan batasan gerakan *roll* dengan *hover* untuk koordinat x pada *leap motion*.

Tabel 6.3 Hasil percobaan gerakan *roll* pada *node js*

Gerakan (*roll*)	Nilai		
	Koordinat x (mm)	Koordinat y (mm)	Koordinat z (mm)
Ke kiri	-71,3866	257,3504	-3,87307
Ke kiri	-140,199	252,9392	-11,3189
Ke kiri	-258,103	241,6976	-12,2746
Ke kiri	-375,393	242,0474	10,17404

Ke kiri	-474,416	233,6288	-5,02401
Ke kanan	42,94622	234,7244	8,830496
Ke kanan	174,9236	248,3298	17,67776
Ke kanan	274,7738	247,7042	25,04316
Ke kanan	355,7488	250,2622	33,10138
Ke kanan	481,778	276,5446	48,26644

Pada tabel 6.4 dibawah juga mengalami hal sama dimana nilai pada sumbu y,z mengalami perubah yang tidak tetap, sedangkan pada sumbu x mengalami perubahan yang tetap dimana nilai negatif digunakan untuk gerakan kekiri *quadcopter* dan nilai positif digunakan untuk gerakan kekanan.

Tabel 6.4 Hasil Percobaan Gerakan *Roll* pada *Labview*

Gerakan (*roll*)	**Nilai**		
	Koordinat x (mm)	**Koordinat y (mm)**	**Koordinat z (mm)**
Ke kiri	-76,0018	230,1202	-13,20609
Ke kiri	-95,8734	229,0711	-12,05477
Ke kiri	-119,4324	228,4255	-9,46930
Ke kiri	-138,1555	227,6838	-5,83306
Ke kiri	-156,8766	227,4462	-3,165680
Ke kanan	48,2209	238,1913	-2,23917
Ke kanan	77,9576	237,5786	0,12014
Ke kanan	125,4302	235,5928	4,11020
Ke kanan	154,4519	233,5892	6,87662
Ke kanan	174,1394	228,8335	7,05372

Pada Tabel 6.5 merupakan nilai koordinat yang dihasilkan gerakan *gaz*. Nilai koordinat x, z mengalami perubahan yang tidak tetap. Untuk nilai koordinat y mengalami perubahan yang tetap dikarenakan untuk gerakan *gaz leap motion* menggunakan sumbu y sebagai tumpuan gerakan tersebut. Sama halnya dengan koordinat yang ada pada program *labview* seperti pada Tabel 6.6 sehingga pada gerakan *gaz* diambil nilai koordinat y untuk pedoman menentukan batasan gerakan *gaz* dengan *hover* untuk koordinat y pada *leap motion.* Gerakan *gaz* merupakan gerakan pada *quadcopter* yang digunakan untuk mengirim perintah pada *quadcopter* untuk bergerak ke atas dan ke bawah dimana koordinat y pada *leap motion* digunakan sebagai acuan untuk pemberian

nilai *input* pada *quadcopter*. Data koordinat juga digunakan sebagai pemberian *input* kecepatan secara otomatis pada *quadcopter*.

Tabel 6.5 Hasil Percobaan Gerakan *Gaz* Pada *node js*

Gerakan (*gaz*)	Nilai		
	Koordinat x (mm)	Koordinat y (mm)	Koordinat z (mm)
Ke bawah	-11,1545	195,7632	-12,5625
Ke bawah	5,020682	128,348	-10,7798
Ke bawah	9,152554	95,48624	-7,80281
Ke bawah	14,03245	62,00168	-5,73005
Ke bawah	21,37834	34,8221	-6,04442
Ke atas	-11,3029	305,4712	-17,7881
Ke atas	-14,3218	431,023	-42,0644
Ke atas	-20,6565	526,7486	-54,6586
Ke atas	-22,5122	604,1308	-65,1726
Ke atas	-15,4397	697,164	-82,4665

Pada tabel 6.6 dibawah juga mengalami hal sama dimana nilai pada sumbu x,z mengalami perubah yang tidak tetap, sedangkan pada sumbu y mengalami perubahan yang tetap dimana nilai hasil dari koordinat dilebih dari 300 mm digunakan sebagai acuan gerakan ke atas dan nilai koordinat dikurang dari 200 digunakan sebagai acuan gerakan ke bawah pada sistem.

Tabel 6.6 Hasil Percobaan Gerakan *Gaz* Pada *Labview*

Gerakan (*gaz*)	Nilai		
	Koordinat x (mm)	Koordinat y (mm)	Koordinat z (mm)
Ke bawah	5,4070	164,5637	-5,10160
Ke bawah	6,9846	154,6336	-6,91322
Ke bawah	9,4909	108,9238	-8,62445
Ke bawah	11,8877	82,0943	-6,39053
Ke bawah	11,1602	51,7061	-10,68784
Ke atas	4,3072	306,8553	-19,37989
Ke atas	11,3072	413,0833	-22,38067

Ke atas	11,2537	510,6063	-32,49928
Ke atas	1,8063	604,1667	-29,75419
Ke atas	9,4210	691,4645	-36,99484

Pada Tabel 6.7 merupakan tabel hasil gerakan *takeoff, hover* dan mendarat pada pemrograman *node js* dan Tabel 6.8 yang dihasilkan dari percobaan pada pemrograman *labview* merupakan nilai koordinat yang dihasilkan pada gerakan *takeoff*, *hover* dan mendarat. Nilai koordinat x, y, z tidak mengalami perubahan yang derastis dikarenakan untuk gerakan *takeoff*, *hover* dan *landing* perbedaannya pada gerakan telapak tangan. Untuk gerakan *takeoff* gerakan telapak tangan yaitu jari-jari membuka lebar dikarenakan perintah untuk *takeoff* pada program *labview* jika *leap motion* mendeteksi lebih dari 3 jari maka *quadcopter* akan otomatis *takeoff* sedangkan pada program *node.js* posisi tangan menggenggam digunakan sebagai acuan untuk gerakan *takeoff*. Untuk gerakan *hover* telapak tangan membuka dan untuk gerakan *landing* pindahkan posisi telapak tangan dari atas *leap motion* atau tutup jari-jari telapak tangan gerakan tersebut digunakan untuk mengurai deteksi jari jika jari yang terdeteksi kurang dari 4 maka otomatis *quadcopter* akan *landing* sedangkan pad program *node.js* telapak tangan bergerak seperti menekan sebuah tombol gerakan tersebut digunakan untuk mengaktifkan gerakan *landing* pada gerakan *takeoff*, *hover*, *landing* diambil nilai koordinat x, y, z, untuk pedoman menentukan batasan gerakan dengan gerakan *pitch*, *roll* dan *gaz*.

Tabel 6.7 Hasil percobaan gerakan *takeoff*, *hover*, *landing* pada *node js*

Gerakan (*takeoff*, *hover*, mendarat)	**Nilai**		
	Koordinat x (mm)	**Koordinat y (mm)**	**Koordinat z (mm)**
Takeoff	5,44241	245,751	-26,1774
Takeoff	6,150441	245,8674	-27,7097
Takeoff	4,084202	247,1518	-28,6956
Takeoff	3,598775	246,9202	-25,1339
Takeoff	4,68569	244,5102	-28,9354
Hover	-1,74629	223,3518	6,001906
Hover	-3,20989	222,9754	5,16865
Hover	-3,63728	222,884	4,931272
Hover	-4,97245	222,645	4,146183
Hover	-5,42672	222,1634	4,568116

Mendarat	-3,10575	247,9068	-0,52181
Mendarat	-5,71061	246,7982	1,312898
Mendarat	-4,56839	248,7728	-0,42281
Mendarat	-4,70385	249,9172	-1,64464
Mendarat	-5,31794	249,5516	-2,69188

Pada tabel 6.8 dibawah juga mengalami hal sama dimana nilai pada sumbu x,y dan z mengalami perubah yang tidak tetap, seperti pada tabel 6.7. pembeda dari gerakan ini terletak pada posisi gerakan tangan pada saat menjalankan *input* pada *quadcopter*.

Tabel 6.8 Hasil percobaan gerakan *takeoff*, *hover*, *landing* pada *labview*

Gerakan (*takeoff*, *hover*, mendarat)	**Nilai**		
	Koordinat x (mm)	**Koordinat y (mm)**	**Koordinat z (mm)**
Takeoff	5,44241	245,751	-26,1774
Takeoff	6,150441	245,8674	-27,7097
Takeoff	4,084202	247,1518	-28,6956
Takeoff	3,598775	246,9202	-25,1339
Takeoff	4,68569	244,5102	-28,9354
Hover	-1,74629	223,3518	6,001906
Hover	-3,20989	222,9754	5,16865
Hover	-3,63728	222,884	4,931272
Hover	-4,97245	222,645	4,146183
Hover	-5,42672	222,1634	4,568116
Mendarat	-3,10575	247,9068	-0,52181
Mendarat	-5,71061	246,7982	1,312898
Mendarat	-4,56839	248,7728	-0,42281
Mendarat	-4,70385	249,9172	-1,64464
Mendarat	-5,31794	249,5516	-2,69188

6.2 Pengujian Ketepatan Gerakan

6.2.1 Tujuan Pengujian

Pada Pengujian ini bertujuan untuk mengetahui apakah sistem berjalan sesuai dengan yang dirancang penulis secara tepat.

6.2.2 Pelaksanaan Pengujian

Pengujian ketepatan gerakan dilakukan sebanyak 5 kali. Setiap intruksi gerakan yang dikirim menuju *quadcopter* dapat dilihat *output* pada *command prompt* pada program *node.js atau nav data* pada *labview* di destop komputer. Setelah mengamati *output* pada komputer juga mengamati apakah *quadcopter* bergerak sesuai dengan intruksi yang pengguna. Intruksi yang diberikan antara lain *roll* ke kanan, *roll* ke kiri, *pitch* ke depan, *pitch* ke belakang, *takeoff*, *hover* dan *landing*.

6.2.3 Langkah-Langkah Pengujian

Berikut merupakan langkah-langkah yang akan dilakukan dalam pengujian:

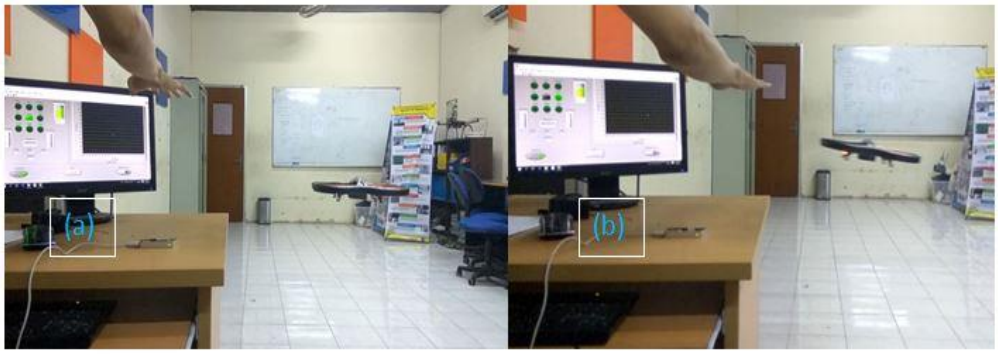
1. *Leap motion* dihubungkan dengan komputer menggunakan kabel *USB.*
2. Membuka aplikasi *leap motion* control panel. Lalu muncul *icon leap motion* pada di pojok kanan bawah destop, jika *icon leap motion* berwarna hijau maka *leap motion* siap digunakan. Jika *icon leap motion* berwarna hitam maka klik kanan pada *icon* tersebut lalu pilih resume *tracking*. Jika *icon* berwarna orange maka *leap motion* perlu dibersihkan permukaannya terlebih dahulu sebelum digunakan.
3. Pasang baterai *quadcopter* dan tunggu sampai *quadcopter* selesai mengkalibrasi. Setelah terdengar suara beeb sebanyak empat kali maka kalibrasi *quadcopter* sudah selesai dan siap digunakan.
4. Menghubungkan komputer dengan *quadcopter* melalui hotspot *Wi-Fi* yang ada pada *quadcopter.*
5. Membuka *command prompt node js* pada komputer lalu menjalankan program "*node* sistemkontrol*quadcopter*.js", sedangkan untuk pengujian pada program *labview* buka aplikasi *labview* buka *project uji* dan jalankan program "*coba uji.vi".*
6. Membuka *command prompt node js* sekali lagi pada komputer lalu setelah terbuka jalankan program "*node* navdata.js".
7. Pengguna memposisikan tangan di atas *leap motion.*
8. Menggerakkan tangan sesuai dengan intruksi yang telah ditentukan yaitu gerakan *roll*, *pitch*, *gaz*, *takeoff*, *hover*, mendarat.
9. Melihat pergerakan *quadcopter* apakah sesuai dengan intruksi yang diberikan.

10. Data navigasi dari *labview* dan pada *command prompt* yang menjalankan navdata.js disalin lalu memindah data tersebut pada *microsoft excel* untuk menganalisa data tersebut.
11. Mengulangi langkah ke-5 sebanyak lima kali untuk setiap percobaan.

6.2.4 Hasil Pengujian

6.2.4.1 Gerakan *Roll*

Pada bagian (a) dan (b) merupakan gerakan yang dilakukan oleh pengguna untuk melakukan pengujian ketepatan gerakan. Pada gambar 6.5 merupakan gambar dari pengujian ketepatan gerakan *roll* kanan. Dimana (a) merupakan posisi awal dan posisi (b) ketika *quadcopter* bergerak.



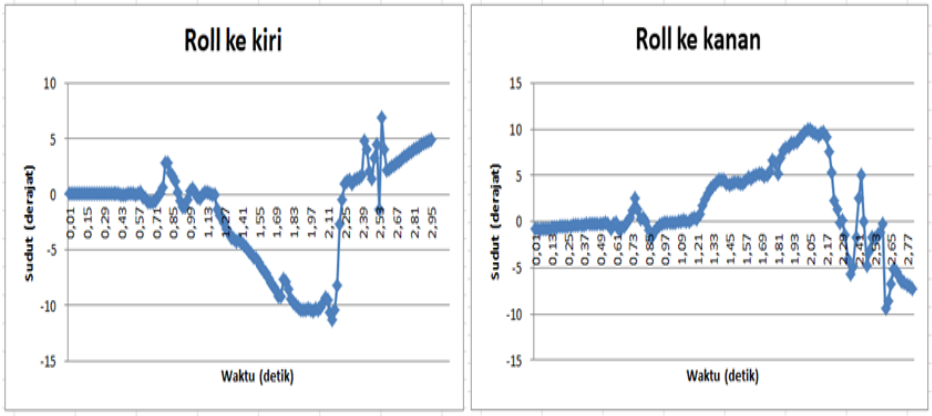
Gambar 6.5 Pengujian gerakan *roll* kanan

Pada Gambar 6.6 dapat dilihat proses melakukan pengujian gerakan *roll* ke kiri. Pada bagian (b) merupakan proses *quadcopter* berada pada posisi stabil awal sebelum diberikan inputan kemudian pada bagian (a) merupakan proses *quadcopter* bergerak setelah diberikan gerakan ke kiri.

Gambar 6.6 Pengujian gerakan kiri

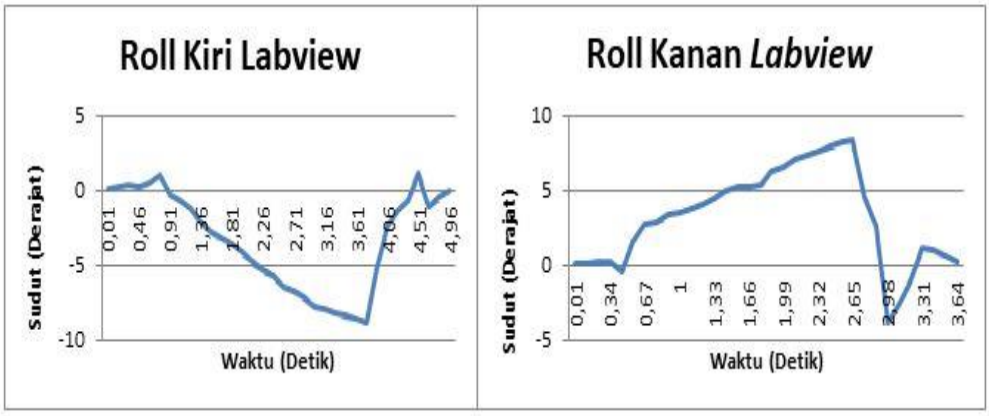
Setelah dilakukan pengujian gerakan *roll* maka dihasilkan grafik seperti pada Gambar 6.7. Pada gambar tersebut terlihat perubahan sudut yang dialami oleh *quadcopter* ketika melakukan gerakan *roll*. Perubahan sudut tersebut cenderung sama antara gerakan *roll* ke kiri dengan gerakan *roll* ke kanan hanya berbeda nilai positif dan negatifnya. Hal ini dikarenakan menggunakan satu koordinat yaitu koordinat y untuk gerakan *roll* ke kiri dan ke kanan. Untuk *roll* ke kiri nilai

perubahan sudut semakin negatif sedangkan untuk gerakan *roll* ke kanan nilai perubahan sudut semakin positif.



Gambar 6.7 Grafik gerakan *roll* pada *node js*

Dari percobaan gerakan *roll* pada pemrograman *labview* menunjukkan hasil yang sama. Dimana perubahan sudut yang terjadi pada gerakan *roll* hampir sama seperti yang dilakukan di program *node js*. Ketika gerakan kekanan sudut *quadcopter* akan bernilai positif dan ketikan *quadcopter* bergerak kekiri maka sudut yang dihasilkan akan bernilai negatif. Berikut grafik gerakan *roll* pada pemrograman *labview* terlihat pada gambar 6.8. Dari data grafik pada gambar 6.8 dibawah juga bisa dilihat bahwa gerakan *roll* yang dijalankan pada pemrograman *labview* lebih stabil dibandingkan dengan menggunakan bahasa pemrograman *node js*, dimana masih terdapat *noise* pada gerakannya.



Gambar 6.8 Grafik gerakan *roll* pada *labview*

6.2.4.2 Gerakan *Pitch*

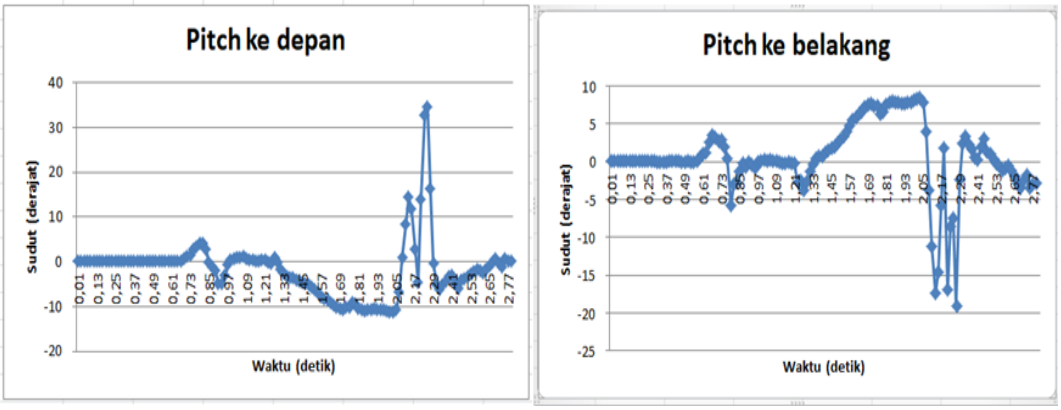
Pada Gambar 6.9 dapat dilihat proses melakukan pengujian gerakan *pitch* ke depan. Pada bagian (a) merupakan posisi awal dari yaitu posisi hover kemudian pada posisi ini quadcopter berada pada posisi stabil sampai pengguna memberikan input untuk gerakan. Kemudian pengguna memberikan inputan berupa gerakan maju pada sistem, setelah mendapatkan inputan quadcopter langsung menjalankan inputan sesuai dengan input gerakan pengguna seperti pada gamabr 6.9 bagian (b) yang menunjukkan quadcopter mulai bergerak maju.

Gambar 6.9 Pengujian gerakan *pitch* ke depan

Pada Gambar 6.10 dapat dilihat proses melakukan pengujian gerakan *pitch* ke belakang. Pada bagian (a) merupakan gerakan awal sebelum gerakan mundur ke belakang quadcopter berada pada posisi hover, kemudian setelah pengguna memberikan input berupa gerakan mundur maka sistem langsung menjalankan dan quadcopter langsung bergerak mundur dari posisi sebelumnya seperti pada gambar 6.10 pada bagian (b).

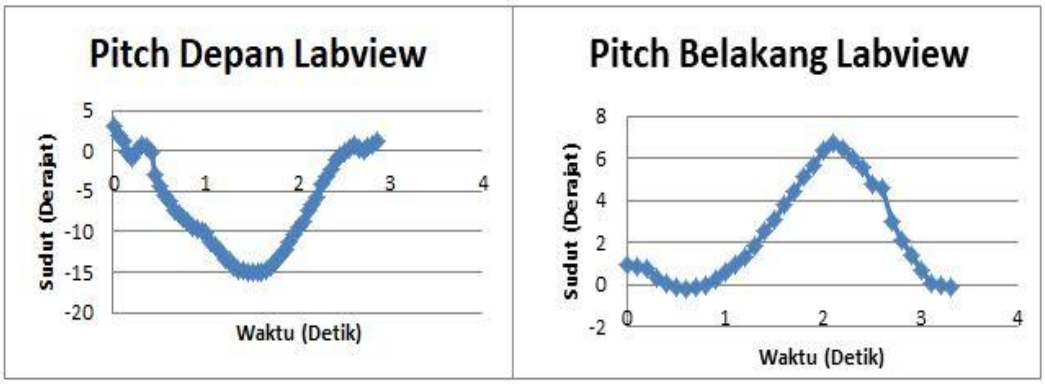
Gambar 6.10 Pengujian gerakan *pitch* ke belakang

Setelah dilakukan pengujian gerakan *pitch* maka dihasilkan grafik seperti pada Gambar 6.11. Pada gambar tersebut terlihat perubahan sudut yang dialami oleh *quadcopter* ketika melakukan gerakan *pitch*. Perubahan sudut tersebut cenderung sama antara gerakan *pitch* ke depan dengan gerakan *pitch* ke belakang hanya berbeda nilai positif dan negatifnya. Hal ini dikarenakan menggunakan satu koordinat yaitu koordinat z untuk gerakan *pitch* ke depan dan ke belakang. Untuk *pitch* ke depan nilai perubahan sudut semakin negatif sedangkan untuk gerakan *pitch* ke belakang nilai perubahan sudut semakin positif.



Gambar 6.11 Grafik gerakan *pitch* pada *node js*

Percobaan gerakan *pitch* pada pemrogramaan *labview* menghasilkan gerakan yang lebih stabil dibandingkan dengan percobaan gerakan *pitch* pada pemrograman *node js*. Hasil percobaan *pitch* pada *labview* seperti pada gambar 6.12. Dari grafik pada gambar 6.12 dibawah dapat dilihat ketika *input pitch* dijalankan maka *quadcopter* akan berjalan sesuai *input* dan ketika *input* selesai dijalankan kan *quadcopter* akan kembali ke posisi *hover* atau posisi stabil *quadcopter*. Hasil tersebut menunjukkan gerakan *pitch* lebih stabil dijalankan ketika menggunakan bahasa pemrograman *labview* dibandingkan *node js* yang masih terdapat *noise* pada gerakan.



Gambar 6.12 Grafik gerakan *pitch* pada *labview*

6.2.4.3 Gerakan *Gaz*

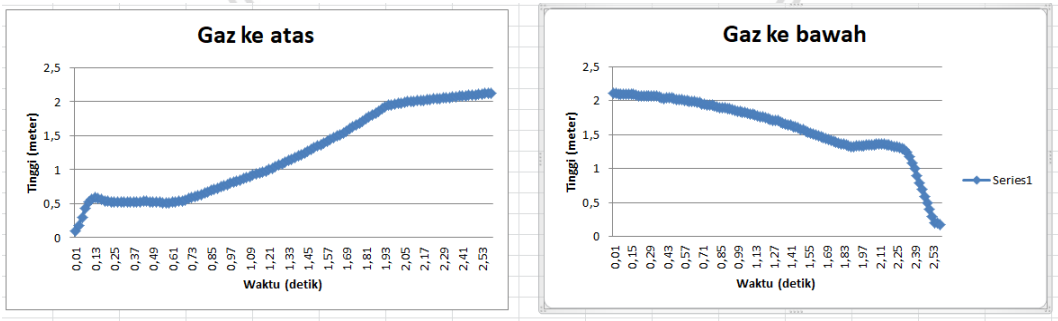
Pada Gambar 6.13 dapat dilihat proses melakukan pengujian gerakan *gaz* ke atas. Pada bagian (a) merupakan gerakan posisi awal *quadcopte*r kemudian setelah diberikan input gerakan *quadcopter* bergerak ke atas seperti yang terlihat pada gambar 6.13 bagian (b).

Pada Gambar 6.14 dapat dilihat proses melakukan pengujian gerakan *gaz* ke bawah. Pada bagian (a) merupakan bagian terakhir input gerakan dilakukan yaitu ke gerakan ke atas kemudian dilakukan input gerakan ke bawah seperti gambar 6.14 bagian (b) dari gambar tersebut terlihat *quadcopter* turun dari posisi awal di posisi atas.

Gambar 6.13 Percobaan gerakan *gaz* atas

Gambar 6.14 Percobaan gerakan *gaz* bawah

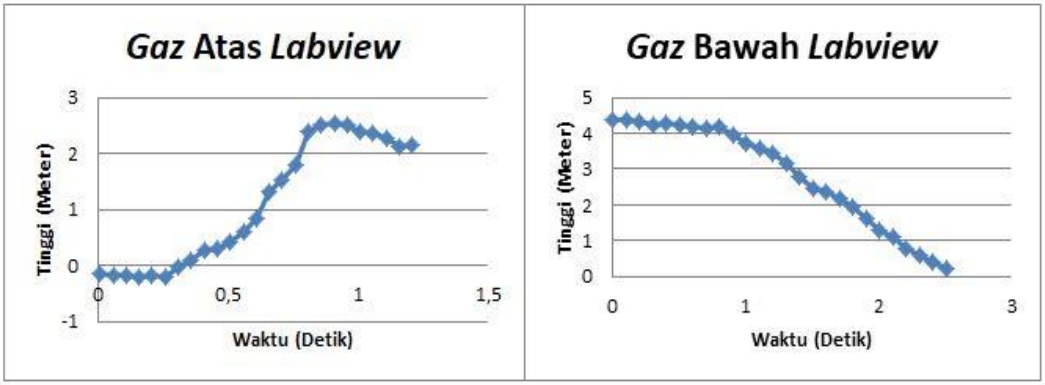
Setelah dilakukan pengujian gerakan *gaz* maka dihasilkan grafik seperti pada Gambar 6.15. Pada gambar tersebut terlihat perubahan ketinggian yang dialami oleh *quadcopter* ketika melakukan gerakan *gaz*. Untuk *gaz* ke atas nilai ketinggian semakin bertambah sedangkan untuk gerakan *gaz* ke bawah nilai ketinggian semakin mengecil.



Gambar 6.15 Grafik Gerakan *gaz* pada *node js*

Sedangkan percobaan gerakan *gaz* pada *labview* menghasilkan data seperti pada gambar 6.16. Dari hasil data percobaan pada gambar 6.16 dapat kita amati bahwa hasil yang ditunjukkan hampir sama dengan gerakan yang dihasilkan pada pemrograman *node js*. Dimana ketika *input* gerakan *gaz* atas dijalankan maka *quadcopter* akan naik semakin tinggi, begitu sebaliknya ketika ketika perintah *gaz*

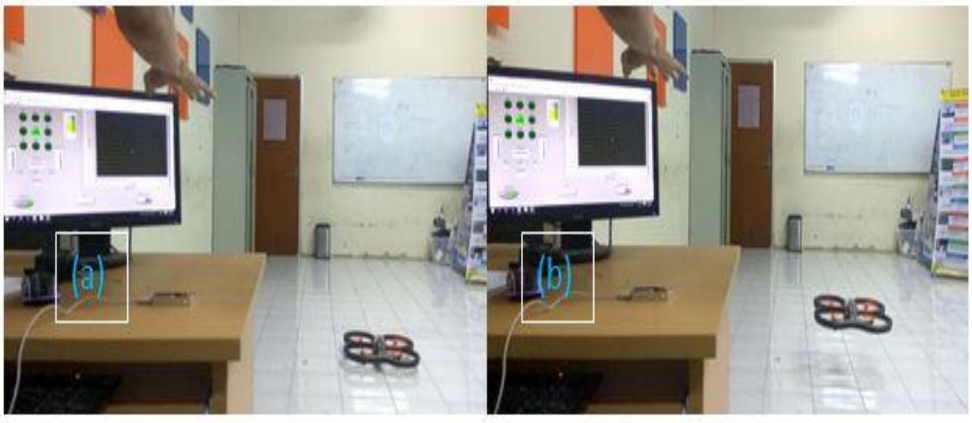
bawah dijalankan maka *quadcopter* akan turun dan ketinggian *quadcpoter* pada saat terbang akan berkurang. Dari data grafik yang dihasilkan dari percobaan gerakan *gaz* pada pemrograman *labview* dapat dilihat untuk gerakan ke bawah lebih stabil dibandingkan dengan percobaan gerakan ke bawah pada pemrograman *node js* sedangkan untuk gerakan gerakan ke atas hampir sama respon *quadcopter* pada pemrograman *node js* maupun *labview*.



Gambar 6.16 Grafik gerakan *gaz* pada *labview*

6.2.4.4 *Gerakan Takeoff, Hover* dan Mendarat

Pada Gambar 6.17 dapat dilihat proses melakukan pengujian gerakan *takeoff*. Pada bagian (a) merupaka posisi awal *quadcopter* belum terbang kemudian setelah pengguna memberikan input gerakan *takeoff* pada sistem maka otomatis sistem akan menjalankan .

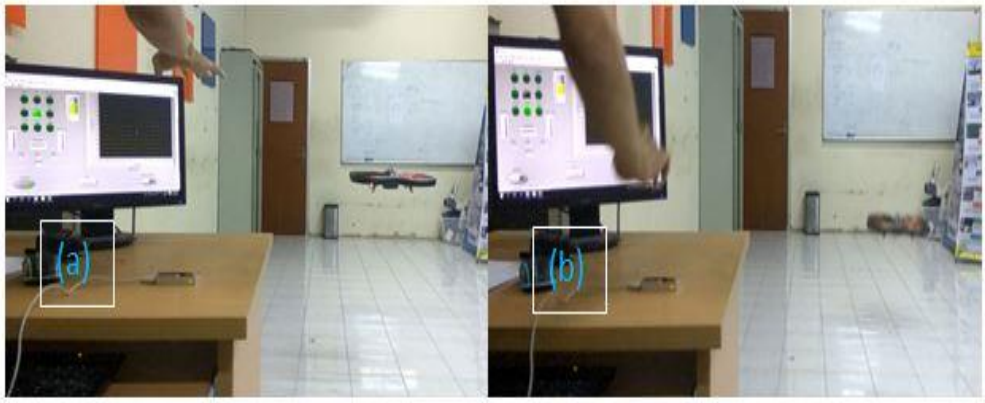


Gambar 6.17 Percobaan gerakan *takeoff*

Pada Gambar 6.18 dapat dilihat proses melakukan pengujian gerakan *hover*. Pada bagian (a) merupak posisi awal sebelum *quadcopter* dikasih perintah. Kemudian setelah pengguna memberikan *input* gerakan *hover* pada sistem maka otomatis *quadcopter* akan terbang dan stabil di ketinggian tertentu *quadcopter* akan terbang stabil sampai pengguna mengirimkan input gerakan kembali pada *quadcopter* seperti yang terlihat pada gambar 6.18 bagian (b) *quadcopter* pada posisi *hover* dimana *quadcopter* terbang dengan stabil.

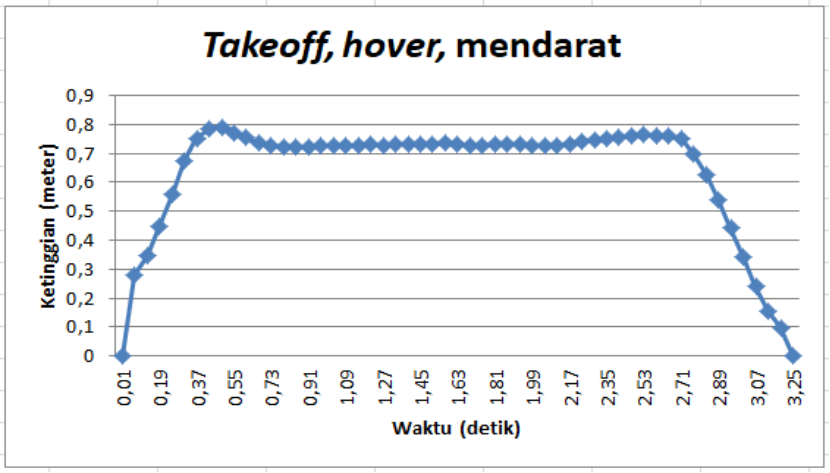
Gambar 6.18 Percobaan gerakan *hover*

Pada Gambar 6.19 dapat dilihat proses melakukan pengujian gerakan mendarat. Pada bagian (a) gerakan yang dilakukan sebelumnya yaitu gerakan hover kemudian pengguna memberikan input pada sistem berupa gerakan mendarat dan quadcopter akan langsung mendarat sesuai perintah dari pengguna seperti yang terlihat pada gambar 6.19 dimana tangan dari mulai beranjak meninggalkan posisi dari leap motion kemudian diikuti gerakakn dari quadcopter yang otomatis mendarat.



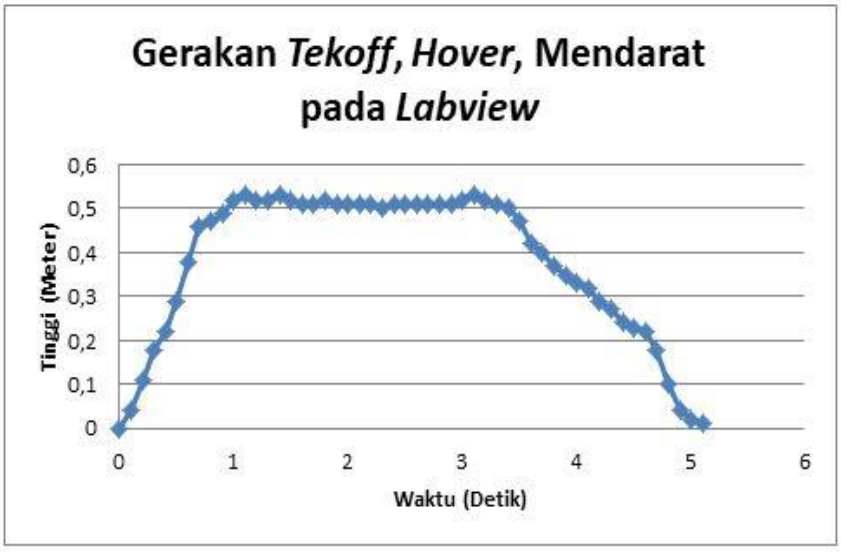
Gambar 6.19 Percobaan gerakan mendarat

Setelah dilakukan pengujian gerakan *takeoff, hover* dan mendarat maka dihasilkan grafik seperti pada Gambar 6.20 dibawah data tersebut merupakan hasil percobaan gerakan yang dilakukan pada pemrograman *node js*. Kemudian dilakukan percobaan gerakan yang sama pada pemrograman *labview* dan menghasilkan data seperti pada gambar 6.21. Pada gambar tersebut terlihat perubahan ketinggian yang dialami oleh *quadcopter* ketika melakukan gerakan *takeoff, hover* kemudian mendarat. Untuk *takeoff* nilai ketinggian semakin bertambah sedangkan untuk gerakan *hover* stabil di ketinggian tertentu dan diikuti gerakan mendarat yang menandakan ketinggian *quadcopter* semakin berkurang.



Gambar 6.20 Grafik gerakan *takeoff*, *hover* dan mendarat *node js*

Sedangkan percobaan gerakan *takeoff*, *hover* dan mendarat pada *labview* menghasilkan data seperti pada gambar 2.21. Hasil yang ditunjukkan hampir dimana ketikan gerakan *takeoff* diberikan oleh pengguna sebagai *input* maka sistem akan menjalankan perintah dan *quadcopter* akan otomatis terbang dapat dilihat pada kedua grafik ketika *takeoff* dijalankan ketinggian *quadcopter* akan berubah sampai posisi *quadcopter* berada pada posisi *hover*. Posisi *hover* adalah posisi dimana *quadcopter* berada pada posisi stabil untuk ketinggian tidak akan mengalami perubahan secara drastis hal tersebut dikarenakan posisi *quadcopter* berada pada posisi stabil *quadcopter*, kecuali *quadcopter* diberikan *input* gerakan maka ketinggian akan berubah, misalkan diberikan perintah mendarat. Perintah mendarat yang diberikan pengguna akan mempengaruhi ketinggian *quadcopter* dimana ketinggian *quadcopter* akan berkurang dari posisi *hover* atau posisi stabil *quadcopter*.



Gambar 6.21 Grafik gerakan *takeoff*, *hover* dan mendarat pada *labview*

Setelah pengguna *node js* dan pengguna *labview* melakukan percobaan sebanyak lima kali maka dihasilkan seperti pada Tabel 6.6 dan Tabel 6.7. Untuk

Tabel 6.6 merupakan hasil pengujian pengguna *node js* dan untuk Tabel 6.7 merupakan hasil pengujian pengguna labview. Simbol (√) merupakan tanda jika percobaan berhasil, simbol (×) merupakan tanda jika percobaan tidak berhasil.

Tabel 6.9 Hasil pengujian ketepatan gerakan pengguna *node js*

Gerakan	Percobaan ke				
	1	2	3	4	5
Pitch ke depan	√	√	√	√	√
Pitch ke belakang	√	√	√	√	√
Roll ke kiri	√	√	√	√	√
Roll ke kanan	√	√	√	√	√
Gaz ke atas	√	√	√	√	√
Gaz ke bawah	√	√	√	√	√
Takeoff	√	√	√	√	√
Hover	√	√	√	√	√
Mendarat	√	√	√	√	√

Tabel 6.10 Hasil pengujian ketepatan gerakan pengguna *labview*

Gerakan	Percobaan ke				
	1	2	3	4	5
Pitch ke depan	√	√	√	√	√
Pitch ke belakang	√	√	√	√	√
Roll ke kiri	√	√	√	√	√
Roll ke kanan	√	√	√	√	√
Gaz ke atas	√	√	√	√	√
Gaz ke bawah	√	√	√	√	√
Takeoff	√	√	√	√	√
Hover	√	√	√	√	√
Mendarat	√	√	√	√	√

6.2.5 Analisis Hasil Pengujian

Dari pengujian yang dilakukan dihasilkan berupa keseluruhan pengguna dapat mengendalikan *quadcopter* dengan benar dari kelima kali percobaan. Sehingga dapat dihitung persentase ketepatan gerakan dengan Persamaan 6.1

$$\text{Persentase ketepatan} = \frac{\text{Jumlah gerakan benar}}{\text{Jumlah pengujian}} \text{ X } 100\% \quad (6.1)$$

Pada Tabel 6.8 dapat dilihat sistem ini mampu menghasilkan ketepatan menghasilkan gerakan 100% pada kedua pengguna pada pemrograman *node js* dan *labview*.

Tabel 6.11 Hasil Persentase ketapatan gerakan

Gerakan	Jumlah pengujian	Jumlah gerakan yang benar (Pengguna *node js*)	Jumlah gerakan yang benar (Pengguna *labview*)	Persentase ketepatan
Pitch ke depan	5	5	5	100%
Pitch ke belakang	5	5	5	100%
Roll ke kiri	5	5	5	100%
Roll ke kanan	5	5	5	100%
Gaz ke atas	5	5	5	100%
Gaz ke bawah	5	5	5	100%
Takeoff	5	5	5	100%
Hover	5	5	5	100%
Mendarat	5	5	5	100%

6.3 Pengujian Fungsi Kecepatan

6.3.1 Tujuan Pengujian

Pengujian bertujuan untuk mengetahui perancangan dari fungsi kecepatan apakah berjalan sesuai yang dirancang penulis atau tidak.

6.3.2 Pelaksanaan Pengujian

Pengujian ini dilakukan dengan mengendalikan *quadcopter* dari titik awal menuju titik akhir. Pada saat proses perjalanan dari titik awal menuju titik akhir, gerakan yang digunakan beracuan pada koordinat x, y, z pada *quadcopter*.Untuk pengujian kecepatan pada koordinat x digunakan gerakan *pitch* ke depan, setelah itu pada koordinat y digunakan gerakan *roll* kanan, lalu pada koordinat z

digunakan gerakan *gaz* ke atas. Untuk setiap gerakan yang diuji dilakukan perubahan gerak tangan dengan konstan contoh saat gerakan *pitch* ke depan tangan pengguna bergerak maju secara konstan sampai *quadcopter* berada pada titik akhir. Data yang diambil dari pengujian tersebut yaitu nilai kecepatan pada *quadcopter* yang terdapat pada data navigasi yang dikirim langsung oleh *quadcopter* dan nilai *input* kecepatan dari pergerakan tangan pengguna berdasarkan pada sub bab 5.4 yaitu *Vpitch*, *Vroll*, *Vgaz* menyesusaikan dengan gerakan yang dilakukan. Berdasarkan dokumentasi pihak *Parrot AR Drone* kecepatan *quadcopter* memiliki satuan mm/s. Namun pada pengujian ini diubah menjadi m/s menyesuaikan dengan kecepatan yang diberikan dari pengguna. Hal ini dikarenakan selisih nilai yang terpaut jauh antara kecepatan *quadcopter* yang dapat mencapai nilai diatas 1 mm/s sedangkan masukan dari pengguna maksimal hanya bernilai 1 saja dalam satuan *float* sesuai pada format pengiriman *AT Command*. Kemudian data tersebut akan disimpan dalam *file* berekstensi .xls untuk dianalisis hasilnya.

6.3.3 Langkah-Langkah Pengujian

Berikut merupakan langkah-langkah yang akan dilakukan dalam pengujian:

1. Hubungkan *leap motion* dengan komputer menggunakan *USB*.
2. Membuka aplikasi *leap motion* control panel. Setelah itu muncul *icon leap motion* pada di pojok kanan bawah destop, jika *icon leap motion* berwarna hijau maka *leap motion* siap digunakan. Jika *icon leap motion* berwarna hitam maka klik kanan pada *icon* tersebut lalu pilih resume *tracking*. Jika *icon* berwarna orange maka *leap motion* perlu dibersihkan permukaannya terlebih dahulu sebelum digunakan.
3. Memasang baterai pada *quadcopter* dan menunggu sampai *quadcopter* selesai mengkalibrasi. Setelah terdengar suara beeb sebanyak empat kali maka kalibrasi *quadcopter* sudah selesai dan siap digunakan.
4. Menghubungkan komputer dengan *quadcopter* melalui *Wi-Fi* yang ada pada *quadcopter.*
5. Membuka *command prompt node js* pada computer lalu menjalankan program "*node* sistemkontrol*quadcopter*.js", sedangkan untuk pengujian pada program *labview* buka aplikasi *labview* buka *project uji* dan jalankan program "*coba uji.vi".*
6. Membuka *command prompt node js* sekali lagi pada komputer lalu setelah terbuka jalankan program "*node* navdata.js".
7. Pengguna memposisikan tangan di atas *leap motion.*
8. Menggerakkan tangan sesuai dengan intruksi yang telah ditentukan yaitu gerakan *roll*, *pitch*, *gaz*. Contoh gerakan *pitch* maka pengguna akan menggerakkan tangan mulai terangkat sedikit dari posisi *hover* sampai dengan terangkat dengan tinggi secara berkelanjutan, begitu pula dengan instruksi gerakan lainnya.

9. Pindah data navigasi dari *nav data* pada *labview* dan *command prompt* yang menjalankan perintah "*node* navdata.js" menuju *microsoft excel* untuk dianalisa.
10. Mengulangi tahap ke-5 pengujian sebanyak lima kali.

6.3.4 Hasil Pengujian

Setalah dilakukan pengujian akan didapat data kecepatan *input* dari tangan dan kecepatan dari *quadcopter*. Berikut penjelasan lebih lengkapnya:

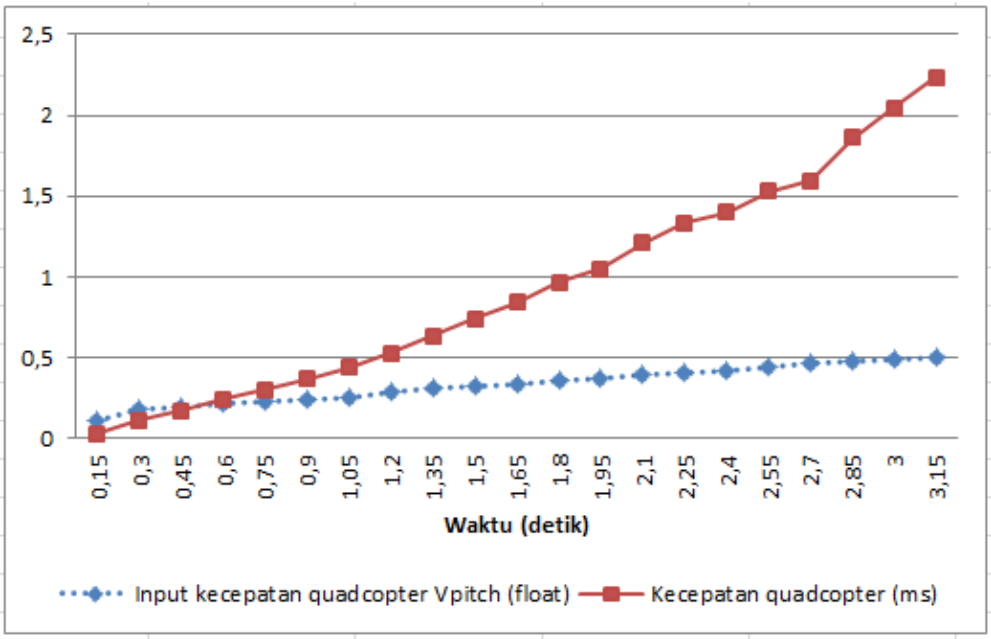
6.3.4.1 Kecepatan *Pitch*

Nilai pada Tabel 6.12 dihasilkan sesuai dengan Persamaan 5.2 dalam kondisi tangan pengguna bergerak ke depan dan bergerak ke belakang. Pada saat tangan pengguna bergerak semakin ke depan maka nilai dari *Vpitch* akan bertambah dan pada saat tangan pengguna bergerak semakin ke belakang maka nilai dari *Vpitch* akan bertambah juga.

Tabel 6.12 Perhitungan *vpitch* pada *node js*

Gerakan	*PositionZ*	*Vpitch*
Ke depan	-366,637	-0,916593
Ke depan	-286,688	-0,71672
Ke depan	-207,46	-0,51865
Ke depan	-123,226	-0,308065
Ke depan	-48,2572	-0,120643
Ke belakang	74,8221	0,187055
Ke belakang	121,198	0,302995
Ke belakang	201,966	0,504915
Ke belakang	284,965	0,712413
Ke belakang	365,91	0,914775

Lalu dalam gerakan ini didapatkan nilai dari kelima kali percobaan dan dirata-rata. Hasilnya dapat dilihat pada Tabel 6.12. Nilai gerakan *pitch* selengkapnya terdapat pada lampiran. Dari nilai pada tabel didapatkan grafik perbandingan antara kecepatan *quadcopter* dengan kecepatan yang diberikan pengguna terhadap *quadcopter* terlihat pada Gambar 6.21.



Gambar 6.22 Grafik pengaruh *input* kecepatan pitch pada *node js*

Tabel 6.13 Nilai kecepatan *input* dan kecepatan *quadcopter* pada *node js*

Waktu (Detik)	*Input* Kecepatan *Vpitch* (*float*)	Kecepatan *Quadcopter* Vx (m/s)
0,15	0,1149694	0,030799786
0,3	0,17893914	0,115816949
0,45	0,19929675	0,174738847
0,6	0,2184414	0,241628899
0,75	0,22880805	0,303187512
0,9	0,24222705	0,367505505
1,05	0,25762475	0,441977954
1,2	0,285013	0,531090222
1,35	0,3084505	0,63306051
1,5	0,323617	0,739113232
1,65	0,33988	0,840517236
1,8	0,3638115	0,967685608
1,95	0,370537	1,051508423
2,1	0,391718	1,207352661
2,25	0,4084265	1,332787622
2,4	0,42479	1,396609253
2,55	0,442995	1,526743774

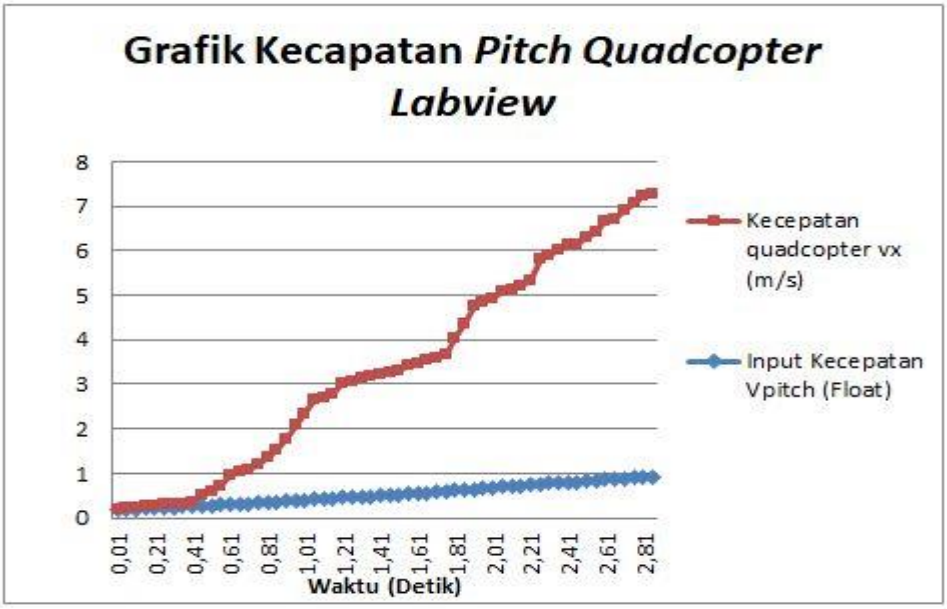
2,7	0,4607705	1,59179978
2,85	0,4818925	1,860204321
3	0,4947375	2,047371631
3,15	0,502619	2,232639404

Pengujian kecepatan gerakan *pitch* pada *labview* hampir sama hasil yang didapat tidak jauh berbeda dimana perancangna kecepatan berjalan sesuai dengan yang dirancang peneliti. Dapat dilihat seperti pada tabel 6.14 dimana kecepatan yang dihasilkan pada gerakan *pitch* di *labview* hampir sempurna dimana kecepatan dimana kecepatan kebelakang hampir mendekati kecepatan 1 yaitu kecepatan maksimal pada *quadcopter*.

Tabel 6.14 Perhitungan *vpitch* pada *labview*

Gerakan	***Position*Z**	***Vpitch***
Ke depan	-370,868408	-0,927171
Ke depan	-307,087372	-0,767718
Ke depan	-213,234207	-0,533086
Ke depan	-133,353104	-0,333383
Ke depan	-41,310734	-0,103277
Ke belakang	70,101425	0,175254
Ke belakang	131,254593	0,328136
Ke belakang	206,447830	0,516120
Ke belakang	297,677673	0,744194
Ke belakang	397,318115	0,993295

Kecepatan *quadcopter* yang dihasilkan pada percobaan gerakan *pitch* di *labview* akan tunjukkan pada gambar 6.22 dibawah. Dari data tersebut dapat dilihat bahwa data yang d*input*kan pengguna berjalan sesuai dengan yang dirancang peneliti. Data kecepatan pada gerakan *pitch quadcopter* di *labview* menunjukkan hasil yang sama dengan gerakan *pitch quadcopter* yang dicoba menggunakan bahasa pemrograman *node js*. Dimana kecepatan berbanding lurus dengan *input* yang diberikan pengguna. Seperti yang terlihat dari gambar 6.22 dimana ketika pengguna memberikan *input* maka *quadcopter* akan langsung merespon. Kecepatan yang dihasilkan yaitu ketika pengguna memberika *input* kecepatan kecil maka *quadcopter* berjalan pelan kemudian jika pengguna memberikan *input* semakin besar maka *quadcopter* akan berjalan semakin cepat.



Gambar 6.23 Grafik gerakan *pitch* pada *labview*

Tabel 6.15 merupakan data dari kecepatan *quadcopter* berserta *input* dari pengguna pada *labview*.

Tabel 6.15 Kecapatan *quadcopter* pada *labview*

Waktu (Detik)	*Input* Kecepatan *Vpitch* (*float*)	Kecepatan *Quadcopter* Vx (m/s)
0,01	0,176949	0,01665704
0,21	0,197615	0,03405968
0,41	0,230193	0,05839295
0,61	0,270882	0,22718075
0,81	0,310607	0,72047592
1,01	0,373373	1,39056046
1,21	0,414731	2,24511215
1,41	0,453899	2,56644226
1,61	0,50730	2,76627960
1,81	0,531709	2,89534027
2,01	0,61204	3,41369263
2,21	0,647719	4,12253479
2,41	0,712787	4,42623291
2,61	0,782196	5,2472229
2,81	0,884835	6,00613037

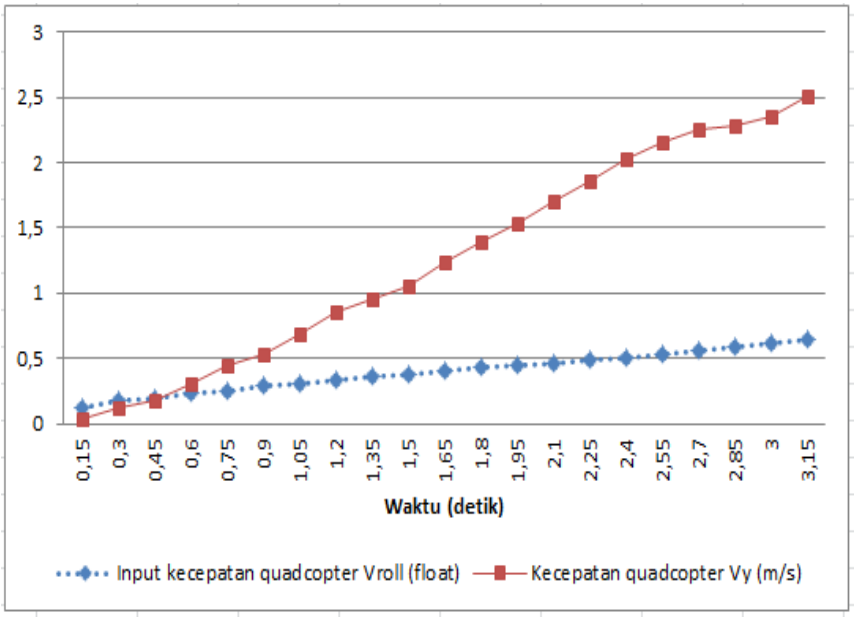
6.3.4.2 Kecepatan *Roll*

Nilai pada Tabel 6.16 dihasilkan sesuai dengan Persamaan 5.2 dalam kondisi tangan pengguna bergerak ke kanan dan bergerak ke kiri. Pada saat tangan pengguna bergerak semakin ke kanan maka nilai dari *Vroll* akan bertambah dan pada saat tangan pengguna bergerak semakin ke kiri maka nilai dari *Vroll* akan bertambah juga.

Lalu dalam gerakan ini didapatkan nilai dari kelima kali percobaan dan dirata-rata. Hasilnya dapat dilihat pada Tabel 6.17. Nilai gerakan *roll* selengkapnya terdapat pada lampiraan. Dari nilai pada tabel didapatkan grafik perbandingan antara kecepatan *quadcopter* dengan kecepatan yang diberikan pengguna terhadap *quadcopter* terlihat pada Gambar 6.23.

Tabel 6.16 Perhitungan *Vroll Quadcopter*

Gerakan	***Position*X**	***Vroll***
Ke kanan	486,937	0,973874
Ke kanan	353,169	0,706338
Ke kanan	250,107	0,500214
Ke kanan	153,995	0,30799
Ke kanan	67,639	0,135278
Ke kiri	-68,5287	0,137057
Ke kiri	-159,948	0,319896
Ke kiri	-257,964	0,515928
Ke kiri	-356,452	0,712904
Ke kiri	-475,581	0,951162



Gambar 6.24 Pengaruh *input* pengguna terhadap kecepatan *roll node js*

Tabel 6.17 Kecepatan gerakan *roll quadcopter* pada *node js*

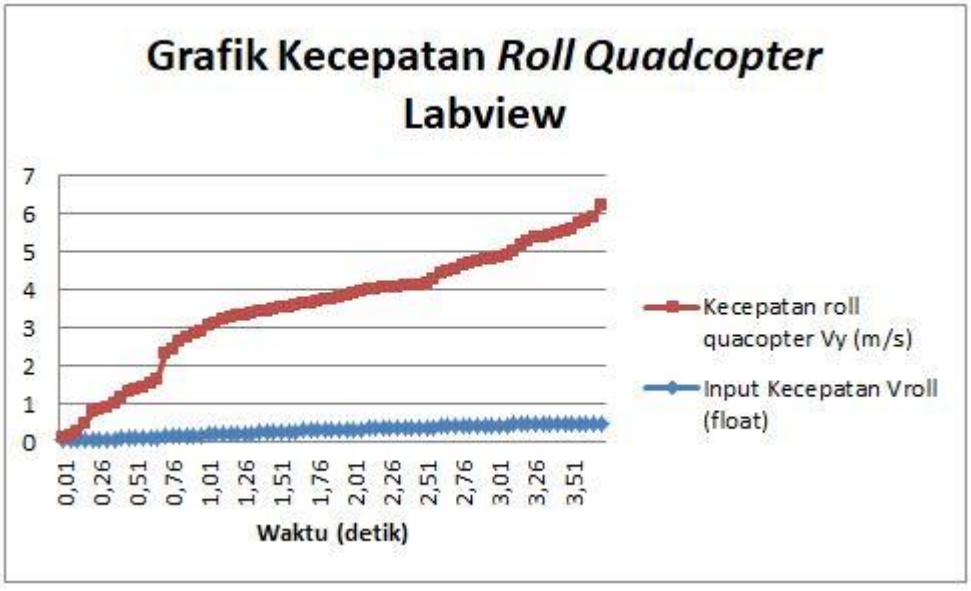
Waktu (Detik)	*Input* Kecepatan *Vroll* (*float*)	Kecepatan *Quadcopter* Vy (m/s)
0,15	0,119284167	0,035799786
0,3	0,178905	0,115916949
0,45	0,19602	0,171738847
0,6	0,226915	0,301187512
0,75	0,253196667	0,439977954
0,9	0,2886	0,535090222
1,05	0,306178333	0,68106051
1,2	0,337585	0,850517236
1,35	0,361243333	0,947685608
1,5	0,378075	1,057508423
1,65	0,401926667	1,230787622
1,8	0,428615	1,399609253
1,95	0,445225	1,536743774
2,1	0,462891667	1,70179978
2,25	0,48234	1,864204321
2,4	0,502619	2,032639404
2,55	0,535225	2,154680667

2,7	0,556446667	2,252208333
2,85	0,580075	2,290171667
3	0,620619	2,359734833
3,15	0,638535	2,507639404

Kemudian untuk pengujian pada kecapatan *roll quadcopter* pada pemrograman *labview*. Dapat dilihat hasil data pada tabel 6.18 yang berisikan perhitungan *vroll* yang digunakan sebagai *input* gerakan *roll* pada *quadcopter.*

Tabel 6.18 Perhitungan *Vroll quadcopter* pada *labview*

Gerakan	*Position*X	*Vroll*
Ke kanan	488,915253	0,977831
Ke kanan	361,805023	0,723610
Ke kanan	258,188507	0,516377
Ke kanan	182,151154	0,364302
Ke kanan	32,057640	0,064115
Ke kiri	-66,576012	-0,133152
Ke kiri	-171,295288	-0,342591
Ke kiri	-283,025421	-0,566051
Ke kiri	-369,844727	-0,739689
Ke kiri	-498,960724	-0,997921



Gambar 6.25 Pengaruh *input* pengguna terhadap kecepatan *roll labview*

Dari gambar 6.25 dapat dilihat percobaan gerakan *roll* pada pemrograman *labview* menunjukkan hasil yang sesuai dengan perancangan dimana ketika *input* dari pengguna semakin besar maka laju *quadcopter* semakin cepat.

Tabel 6.19 *Input* dan *output* kecepatan *quadcopter* gerakan *roll* di *labview*

Waktu (Detik)	*Input* Kecepatan *Vroll* (*float*)	Kecepatan *Quadcopter* Vy (m/s)
0,15	0,119284167	0,035799786
0,3	0,178905	0,115916949
0,45	0,19602	0,171738847
0,6	0,226915	0,301187512
0,75	0,253196667	0,439977954
0,9	0,2886	0,535090222
1,05	0,306178333	0,68106051
1,2	0,337585	0,850517236
1,35	0,361243333	0,947685608
1,5	0,378075	1,057508423
1,65	0,401926667	1,230787622
1,8	0,428615	1,399609253
1,95	0,445225	1,536743774
2,1	0,462891667	1,70179978
2,25	0,48234	1,864204321
2,4	0,502619	2,032639404
2,55	0,535225	2,154680667
2,7	0,556446667	2,252208333
2,85	0,580075	2,290171667
3	0,620619	2,359734833
3,15	0,638535	2,507639404

6.3.4.3 Kecapatan Gerakan *Gaz*

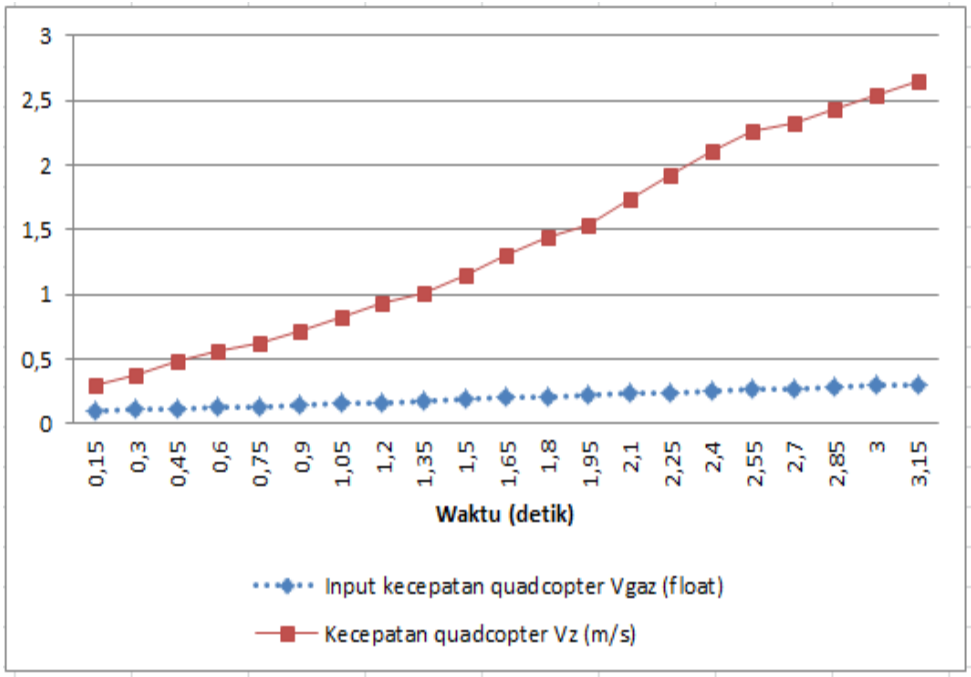
Nilai pada Tabel 6.20 dihasilkan sesuai dengan Persamaan 5.3 dalam kondisi tangan pengguna bergerak ke atas dan bergerak ke bawah. Untuk nilai *Vgaz* yang digunakan adalah *Vgaz* yang dikurangi 0,5 dikarenakan sesuai dengan kebutuhan saat tangan pengguna bergerak semakin ke atas maka nilai dari *Vgaz* akan

bertambah dan pada saat tangan pengguna bergerak semakin ke bawah maka nilai dari *Vgaz* akan bertambah dan nilai maksimal *Vgaz* relatif sama untuk gerakan ke bawah dan ke atas.

Lalu dalam gerakan ini didapatkan nilai dari kelima kali percobaan dan dirata-rata. Hasilnya dapat dilihat pada Tabel 6.18. Nilai gerakan *gaz* selengkapnya terdapat pada lampiran. Dari nilai pada tabel didapatkan grafik perbandingan antara kecepatan *quadcopter* dengan kecepatan yang diberikan pengguna terhadap *quadcopter* terlihat pada Gambar 6.26.

Tabel 6.20 Perhitungan *input Vgaz*

Gerakan	***PositionY***	***Vgaz* dengan dikurangi 0,1**	***Vgaz* dengan dikurangi 0,2**	***Vgaz* dengan dikurangi 0,3**	***Vgaz* dengan dikurangi 0,4**	***Vgaz* dengan dikurangi 0,5**
Ke atas	697,632	0,896617	0,796617	0,696617	0,596617	0,496617
Ke atas	634,13	0,8059	0,7059	0,6059	0,5059	0,4059
Ke atas	560,467	0,700667	0,600667	0,500667	0,400667	0,300667
Ke atas	494,868	0,606954	0,506954	0,406954	0,306954	0,206954
Ke atas	300,663	0,329519	0,229519	0,129519	0,029519	0,07048
Ke bawah	197,917	0,182739	0,082739	0,01726	0,11726	0,21726
Ke bawah	174,615	0,14945	0,04945	0,05055	0,15055	0,25055
Ke bawah	138,339	0,097627	0,002373	0,10237	0,20237	0,30237
Ke bawah	69,9229	0,00011	0,10011	0,20011	0,30011	0,40011
Ke bawah	37,4105	0,046556	0,146556	0,24656	0,34656	0,44656

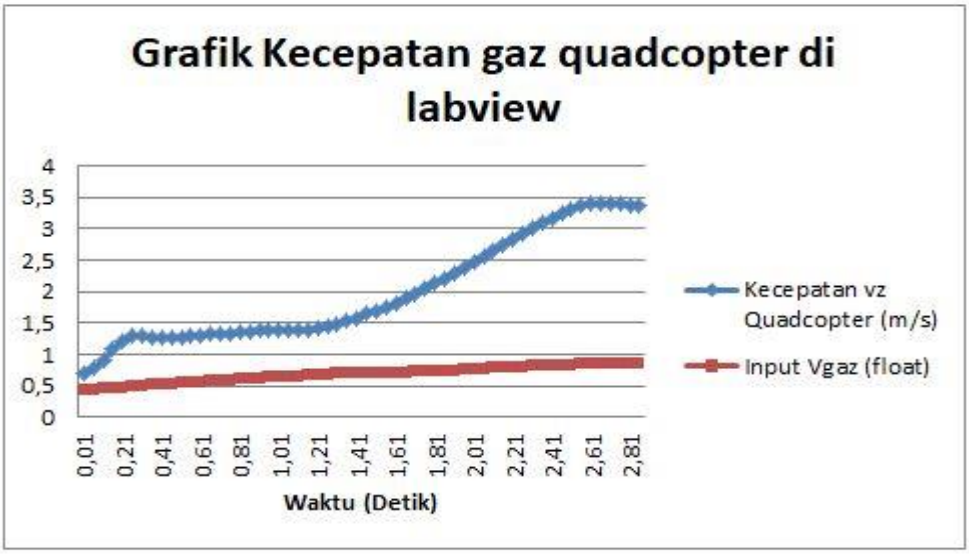


Gambar 6.26 Pengaruh *input* pengguna terhadap kecepatan *gaz node js*

Kemudian untuk pengujian pada kecapatan *gaz quadcopter* pada pemrograman *labview*. Dapat dilihat hasil data pada tabel 6.21 yang berisikan perhitungan *vgaz* yang digunakan sebagai *input* gerakan *roll* pada *quadcopter.*

Tabel 6.21 Perhitungan *input gaz* pada *labview*

Gerakan	*PositionY*	*Vgaz*
Ke Ats	699,453430	0,999219
Ke Atas	557,484314	0,796406
Ke Atas	465,636688	0,665195
Ke Atas	404,420135	0,577743
Ke Atas	304,634216	0,435192
Ke Bawah	199,202179	-0,215425
Ke Bawah	127,841446	-0,317369
Ke Bawah	91,679916	-0,369029
Ke Bawah	67,328949	-0,403816
Ke Bawah	51,573227	-0,426324



Gambar 6.27 Pengaruh *input* pengguna terhadap kecapatan *gaz labview*

Pada gambar 6.26 tersebut dapat dilihat hasil dari percobaan kecepatan *gaz* yang dilakukan pada pemrograman *labview* dimana haasil yang ditunjukkan hasil yang hampir sama dengan percoabaan kecepatan gerakan *gaz* pada pemrograman *node js*. Dimana ketika *input* dari pengguna semakin besar maka kecepatan yang dihasilkan juga akan semakin besar. Perbedaan pada pengujian pada *labview* menghasilkan *input* dari pengguna hampir mendekati *input* maksimal yaitu 1 dibandingkan dengan peercobaan pada *labview* dimana kecepatan *input* pengguna hanya 0,3 sampai 0,4 maksimal *input* pengguna yang diperoleh untuk gerakan *gaz*.

6.3.5 Analisis Hasil pengujian Kecepatan

Pada nilai kecepatan *pitch*, *roll*, *gaz* diambil dalam waktu 3 detik. *Input* kecepatan yang diberikan oleh pengguna pada gerakan *pitch* yaitu 0,502619. Lalu pada gerakan *roll* adalah 0,638535 dan pada gerakan *gaz* yaitu 0,303495. K eseluruhan dari nilai *input* tidak sampai mencapai 1 sesuai dengan perancangan. Jika *input* kecepatan mencapai 1 maka memungkinkan *quadcopter* hilang kendali dan beresiko menabrak.

Berbeda halnya dengan nilai kecepatan *pitch*, *roll* dan *gaz* yang dilakukan di *labview* dimana hasil kecepatan maksimal yang di dapat pada *input* gerakan *pitch* adalah 0,919778. Gerakan *roll* yang dilakukan di *labview input* kecapatan yang didapat adalah 0.517374 dari hasil tersebut lebih besar *input* yang didapat dari percobaan pada pemrograman *node js*. Sedangkan untuk *input* kecepatan gerakan *gaz* yang dilakukan di *labview* menghasilkan kecepatan sebesar 0,86265 dimana hasil tersebut lebih baik dari percobaan yang dilakukan pada pemrograman *node js*.

6.4 Pengujian Respon Sistem

6.4.1 Tujuan Pengujian

Pengujian ini dilakukan untuk mengetahui seberapa baik performa dari sistem kendali yang telah dibuat dengan melihat seberapa besar waktu terjadinya *delay* yang terjadi saat pengguna menggerakkan tangan sampai *quadcopter* dapat bergerak.

6.4.2 Pelaksanaan Pengujian

Pengujian ini dilakukan dengan melihat selisih *frame* antara gerakan tangan pengguna yang telah dilakukan dengan sempurna hingga *quadcopter* dapat bergerak. Untuk mengetahui selisih *frame* tersebut dengan akurat, maka akan dilakukan perekaman video pada saat pengguna mengendalikan *quadcopter*. Hasil dari video tersebut akan dianalisis menggunakan *editor video Adobe Premiere Pro* dengan menghitung selisih *frame* gambar antara gerakan tangan pengguna dengan gerakan *quadcopter.*

6.4.3 Langkah-Langkah Pengujian

Berikut merupakan langkah-langkah yang akan dilakukan dalam pengujian:

1. Menghubungkan *leap motion* dengan komputer menggunakan kabel *USB.*
2. Membuka aplikasi *leap motion* control panel. Setelah itu muncul *icon leap motion* pada di pojok kanan bawah destop, jika *icon leap motion* berwarna hijau maka *leap motion* siap digunakan. Jika *icon leap motion* berwarna hitam maka klik kanan pada *icon* tersebut lalu pilih resume *tracking*. Jika *icon* berwarna oranye maka *leap motion* perlu dibersihkan permukaannya terlebih dahulu sebelum digunakan.
3. Memasang baterai pada *quadcopter* dan menunggu sampai *quadcopter* selesai mengkalibrasi. Setelah terdengar suara beeb sebanyak empat kali maka kalibrasi *quadcopter* sudah selesai dan siap digunakan.
4. Menghubungkan komputer dengan *quadcopter* melalui *Wi-Fi* yang ada pada *quadcopter.*
5. Membuka *command prompt node js* pada computer lalu menjalankan program "*node* sistemkontrol*quadcopter*.js", sedangkan untuk pengujian pada program *labview* buka aplikasi *labview* buka *project uji* dan jalankan program "*coba uji.vi".*
6. Pengguna memposisikan tangan di atas *leap motion.*
7. Menggerakkan tangan sesuai dengan intruksi yang telah ditentukan yaitu gerakan *roll*, *pitch*, *gaz*, *takeoff*, *hover*, mendarat.
8. Ketika pengguna mengendalikan *quadcopter*, keseluruhan gerakan pengguna dan *quadcopter* direkam menggunakan alat rekam.
9. Mengulangi tahap ke-7 pengujian sebanyak lima kali.

10. Setelah mendapatkan video hasil pengujian,video tersebut dibuka pada *adobe premire pro* dan lakukan pengamatan per *frame* video untuk menghasilkan *delay* yang akurat. *Delay* dihitung berdasarkan selisih per *frame* antara pengguna selesai menggerakkan tangan dengan *quadcopter* bergerak sesuai dengan intruksi yang diberikan. Lalu hasil selisih per *frame* dibagi dengan *fps* pada video rekaman hasil pengujian sehingga menghasilkan sebuah *delay.*

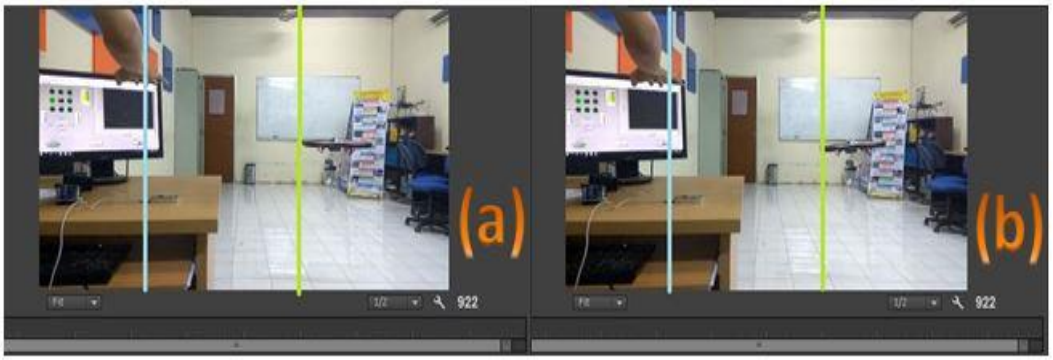
6.4.4 Hasil Pengujian

6.4.4.1 Gerakan *roll*

Pada saat *roll* kanan seperti pada Gambar 6.27 (a) dapat dilihat bahwa *quadcopter* awalnya tepat berada di samping kiri garis kuning pada saat tangan pengguna pada posisis *hover frame* ke 353. Lalu pada Gambar 6.27 (b) diperlihatkan pada *frame* ke 354 tangan pengguna mulai menggeserkan posisi ke kanan dari posisi *hover* dan *quadcopter* mulai melewati garis kuning atau telah melakukan gerakan *roll* ke kanan.

Gambar 6.28 Pengujian Delay Kekanan

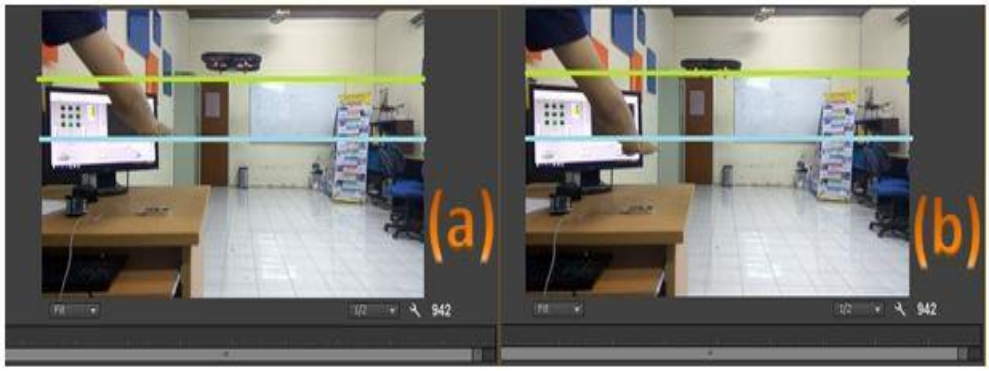
Sedangkan pada Gambar 6.29 diperlihatkan gerakan *roll* ke kiri. Pada Gambar 6.28 (a) pengguna masih melakukan gerakan *hover* pada *frame* ke 698. *Quadcopter* berada tepat di kanan garis kuning. Kemudian pada Gambar 6.28 (b) terlihat *quadcopter* mulai bergerak ke kiri melewati garis kuning pada *frame* ke 699 ketika tangan pengguna bergerak ke kiri menjahui garis biru.



Gambar 6.29 Pengujian Delay Kekiri

6.4.4.2 Gerakan *Gaz*

Pada saat *gaz* ke bawah seperti pada Gambar 6.29 (a) dapat dilihat bahwa *quadcopter* awalnya tepat berada di atas garis hijau pada saat tangan pengguna pada posisi *hover* di atas garis kuning pada *frame* ke 251. Lalu pada Gambar 6.29 (b), diperlihatkan pada *frame* ke 257 tangan pengguna mulai menurunkan posisi dibawah garis biru dan kemudian *quadcopter* mulai melewati garis hijau atau telah melakukan gerakan *Gaz* ke bawah.



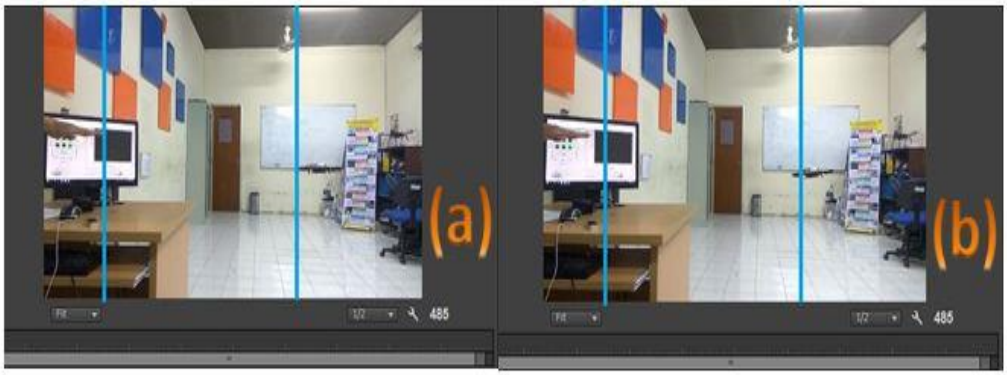
Gambar 6.30 Pengujian Delay Kebawah

Sedangkan pada Gambar 6.31 diperlihatkan gerakan *gaz* ke atas. Pada Gambar 6.30 (a) pengguna masih melakukan gerakan *hover* pada *frame* ke 165. *Quadcopter* berada tepat di atas garis biru. Kemudian pada Gambar 6.30 (b) terlihat *quadcopter* mulai bergerak ke atas menjahui garis biru pada *frame* ke 173 ketika tangan pengguna ber gerak ke atas.

Gambar 6.31 Pengujian Delay Keatas

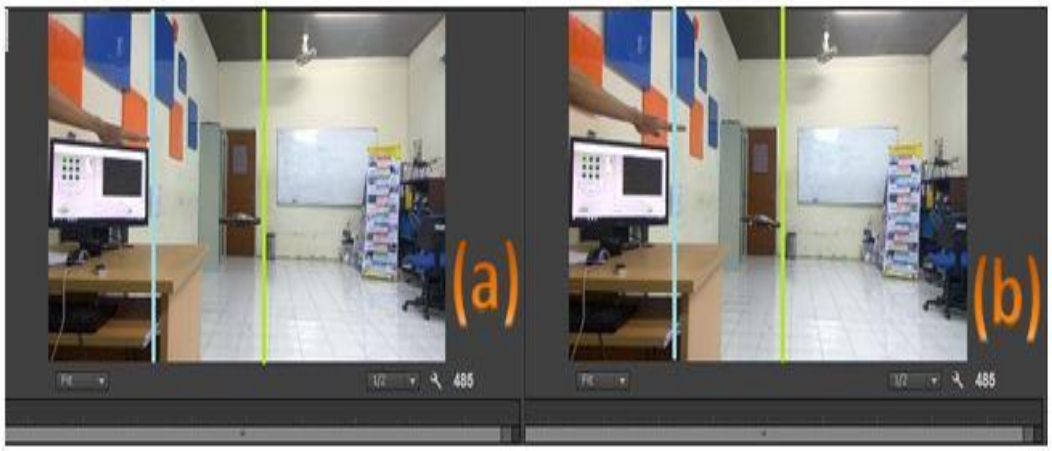
6.4.4.3 Gerakan *Pitch*

Pada saat *pitch* ke belakang seperti pada Gambar 6.31 (a) dapat dilihat bahwa *quadcopter* awalnya tepat berada di samping kanan garis kuning pada saat tangan pengguna pada posisis *hover frame* ke 508. Lalu pada Gambar 6.31 (b), diperlihatkan pada *frame* ke 515 tangan pengguna mulai menggerakkan ke belakang dari garis biru dan *quadcopter* mulai melewati garis kuning atau telah melakukan gerakan *pitch* ke belakang.



Gambar 6.32 Pengujian Delay Kebelakang

Lalu pada Gambar 6.33 diperlihatkan gerakan *pitch* ke depan. Pada Gambar 6.32 (a) pengguna masih dalam posisi gerakan *hover* pada *frame* ke 13. *Quadcopter* berada tepat di kiri garis hijau. Kemudian pada Gambar 6.32 (b) terlihat *quadcopter* mulai bergerak ke kanan dalam posisi miring dan akan melewati garis hijau pada *frame* ke 21 ketika tangan pengguna bergerak ke depan melewati garis biru.



Gambar 6.33 Pengujian Delay Kedepan

6.4.5 Analisis Hasil Pengujian

Setelah melakukan pengujian *delay* pada setiap gerakan pengguna sebanyak lima kali didapat hasil *delay* pada Tabel 6.22 untuk pengujian delay pada *node js* dan tabel 6.23 untuk pengujian *delay* pada pemrograman *labview*. Besar *delay* yang dihasilkan diambil dari selisih per *frame* pergerakan tangan pengguna dengan gerakan *quadcopter* sesuai dengan intruksi yang diberikan pengguna. Kemudian hasil selisih per *frame* dibagi dengan fps pada video rekaman pengujian yaitu 29,97 fps sehingga menghasilkan sebuah *delay*. *Delay* total didapatkan dari nilai rata-rata *delay* dalam lima kali percobaan pada setiap gerakan. Sehingga menghasilkan total *delay* 0,257 detik pada pengujian *delay* pada pemrograman *node js* dan sebesar 0,131 pada pemrograman *labview.*

Tabel 6.22 Hasil d*elay* respon sistem pada *node js*

Gerakan	Pengujian ke-					Rata-rata (detik)
	1	2	3	4	5	
Gaz ke bawah	0,200	0,267	0,200	0,234	0,334	0,247
Gaz ke atas	0,267	0,234	0,200	0,300	0,267	0,254
Roll ke kanan	0,234	0,267	0,300	0,300	0,200	0,260
Roll ke kiri	0,267	0,200	0,234	0,300	0,267	0,254
Pitch ke belakang	0,234	0,267	0,200	0,234	0,334	0,254
Pitch ke depan	0,267	0,267	0,300	0,267	0,267	0,274
Total *Delay*						0,258

Tabel 6.23 Hasil *delay* respon sistem pada *labview*

Gerakan	Pengujian ke-					Rata-rata (detik)
	1	2	3	4	5	
Gaz ke bawah	0,067	0,167	0,100	0,133	0,234	0,140
Gaz ke atas	0,034	0,034	0,134	0,234	0,134	0,114
Roll ke kanan	0,034	0,134	0,134	0,134	0,067	0,221
Roll ke kiri	0,167	0,100	0,067	0,067	0,167	0,114
Pitch ke belakang	0,267	0,134	0,134	0,067	0,067	0,134
Pitch ke depan	0,034	0,034	0,167	0,067	0,034	0,067
Total *Delay*						0,131

BAB 7 PENUTUP

7.1 Kesimpulan

Berdasarkan analisis dari hasil yang telah dilakukan pada penelitian ini, maka dapat diambil beberapa kesimpulan sesuai dengan rumusan masalah yang telah ditentukan sebelumnya yaitu:

1. Menentukan nilai koordinat sebagai nilai batasan setiap gerakan dilakukan pengujian nilai koordinat gerakan kendali *quadcopter*. Pada gerakan ke depan dan ke belakang diambil nilai koordinat sumbu z *leap motion* sebagai nilai batasan gerakan *pitch*. Pada gerakan ke kiri dan ke kanan diambil nilai koordinat sumbu x *leap motion* sebagai nilai batasan gerakan *roll*. Pada gerakan ke bawah dan ke atas diambil nilai koordinat sumbu y *leap motion* sebagai nilai batasan gerakan *gaz*. Pada gerakan *takeoff*, *hover* dan mendarat diambil nilai koordinat x, y, z d sebagai batasan gerakan tersebut. Tidak ada perbedaan pada penentuan nilai koordinat batasan untuk penentuan gerakan pada pemrograman *labview* maupun *node js*. Melainkan butuh dibesarkan lagi untuk batasan-batasan koordinat penentu gerakan pada *labview* hal tersebut dikarenakan pada *labview* sangat responsif ketika posisi tangan. Contohnya ketika akan menjalankan gerakan *roll* usahakan posisi tang pada posisi *roll* jangan condong maju atau terlalu ke atas. Hal tersebut mengakibatkan terdeteksi kedua gerakan meskipun hal tersebut tidak mempengaruhi sistem.
2. Setelah melakukan pengujian ketepatan gerakan dan kecepatan, dihasilkan persentase ketepatan gerakan sebesar 100%. Sehingga dapat diambil kesimpulan bahwa sistem yang telah dibuat telah sesuai dengan harapan dan memiliki nilai akurasi yang tinggi.
3. Dari pengujian *delay* sistem didapat *delay* sebesar 0,258 detik pada pemrograman *node js* dan 0,131 pada pemrograman *labview*. Dari hasil tersebut maka dapat diambil kesimpulan bahwa sistem ini memiliki performa yang sangat baik terlebih sistem mengguna kan pemrograman *labview*.
4. Dari pengujian didapat nilai kecepatan masing-masing gerakan. Kecepatan maksimal untuk gerakan *pitch* yaitu 2,232639404 m/s pada *node js* dan 6,00613037 m/s pada *labview*. Lalu pada gerakan *roll* kecepatan maksimal *quadcopter* adalah 2,507639404 m/s *node js* dan sebesar 2,507639404 m/s pada *labview*. Sedangkan pada gerakan *gaz* kecepatan maksimal *quadcopter* adalah 2,65 m/s pada *node js* dan 3,54 m/s pada *labview* . Pada kecepatan yang dihasilkan oleh *quadcopter* dapat berbanding lurus dengan nilai yang diperoleh dari gerakan pengguna, hal ini menandakan saat *input* dari pengguna semakin besar, maka *quadcopter* akan bertambah cepat dan begitu pula sebaliknya.

5. Dari Hasil keempat data yang telah diuji *labview* menghasilkan proses yang lebih cepat dan responsif dibandingkan *node js*, hal tersebut dikarenakan pembacaan program pada *labview* yang bersifat paralel dari kiri ke kanan pada keseluruhan program yang mengakibatkan proses pembacaan dalam menjalankan program lebih cepat dibandingkan pada *node js* yang bersifat *sequencetial* atau berurutan dari baris pertama hingga akhir yang dapat memakan banyak waktu dalam proses eksekusi program.

7.2 Saran

Terdapat beberapa saran agar sistem ini dapat dikembangkan lebih lanjut di antaranya:

1. Disarankan pada penelitian selanjutkan untuk memperbesar batasan-batasan koordinat penentu gerakan pada *quadcopter*.
2. Saat ini sistem masih dapat mendeteksi gangguan dari tangan pengguna orang lain disekitarnya. Diharapkan pada pengembangan selanjutnya sistem akan tetap dapat dijalankan walaupun ada tangan orang lain di sekitar pengguna.
3. Ditambahkankan gerakan *yaw* pada sistem *labview*.
4. Meminimalisir terjadinya *timeout* pada sistem *labview*.

DAFTAR PUSTAKA

Ardhana, A. B., Setyawan, G. E., & Arwan, I. (2018). Jurnal Pengembangan Teknologi Informasi dan Ilmu Komputer (J-PTIIK). *Sistem Kendali Navigasi Ar Drone Menggunakan Leap Motion* , 2972-2980.

Damar, A. M. (2017). *Pemadam Kebakaran Pakai Drone untuk Permudah Evakuasi*. Dipetik 06 24, 2018, dari https://www.liputan6.com/tekno/read/3074509/pemadam-kebakaran-pakai-drone-untuk-permudah-evakuasi

Hadi, S. W., Setyawan, G. E., & Maulana, R. (2017). Jurnal Pengembangan Teknologi Informasi dan Ilmu Komputer (J-PTIIK. *Sistem Kendali Navigasi Ar.Drone Quadcopter Dengan Prinsip Natural User Interface Menggunakan Microsoft Kinect*, 380-386.

Lilian, C., Setyawan, G. E., & Kurniawan, W. (2018). Jurnal Pengembangan Teknologi Informasi dan Ilmu Komputer (J-PTIIK). *SISTEM PENDARATAN OTOMATIS QUADCOPTER DENGAN PENGOLAHAN CITRA MENGGUNAKAN METODE DOUGLAS PEUCKER*.

Maravall, D., Lope, J. d., & Fuentes, J. P. (2017). *Navigation and Self-Semantic Location of Drones in Indoor Environments by Combining the Visual Bug Algorithm and Entropy-Based Vision*, 11. Dipetik 07 16, 2018, dari https://www.ncbi.nlm.nih.gov/pmc/articles/PMC5581837/

Pallas, F. A., Setyawan, G. E., & Prasetio, B. H. (2017). Jurnal Pengembangan Teknologi Informasi dan Ilmu Komputer (JPTIIK). *Sistem Kendali Navigasi Quadcopter Menggunakan Suara Melalui Smartphone dan Arduino dengan Metode Text Processing*, 732-738.

Pititeeraphab, Y. (2016). Biomedical Engineering (BME-HUST). *Robot-arm control system using LEAP motion controller*, 109-112.

Pribadi, D. A., Jonemaro, E. M., & Setyawan, G. E. (2017). Jurnal Pengembangan Teknologi Informasi dan Ilmu Komputer (JPTIIK). *Implementasi Pengendalian Quadcopter Dengan Prinsip Virtual Reality Menggunakan Google Cardboard* , 1451-1458.

Sarkar, A., Patel , K. A., R.K, G. R., & Capoor , G. K. (2016). *Gesture Control of Drone Using a Motion Controller*, 1.

Skraba, A., Stanovov, V., Semenkin, E., Kolozvari, A., Stojanovic, R., & Kofjac, D. (2015). Embedded Computing (MECO). *Wheelchair Maneuvering Using Leap Motion Controller and Cloud Based Speech Control*, 391-394.

Su´arez Fern´andez, R. A., Sanchez-Lopez, J. L., Sampedro, C., Bavle, H., Molina, M., & Campoy, P. (2016). *Natural User Interfaces for Human-Drone Multi-Modal Interaction. Unmanned Aircraft Systems (ICUAS)*, 1013-1022.

Tolentino, R. E. (2016). *Mimicking the Movements of the Human Hand using Leap Motion Sensor for Different Users*, 35-41.